S7.1

With the PID controller having its proportional and derivative actions in the feedback path, the steady-state error is given in Eq. 7.2:

$$\Delta\theta(\infty)=\lim_{z\to 1}\left(\frac{z-1}{z}\Delta\theta(z)\right)=R^*\frac{p}{i}.$$

With the proportional gain relocated into the direct path, the steady-state error is obtained in Eq. 7.5:

$$\Delta\theta(\infty)=\lim_{z\to 1}\left(\frac{z-1}{z}\Delta\theta(z)\right)=\lim_{z\to 1}\left(\frac{z-1}{z}\ \frac{2d}{2i}\ R^*\right)=0.$$

S7.2

The Matlab command file *P7_2cmd* initializes the parameters, runs the two model files, and plots the resulting traces. The output error obtained with the proportional gain in the direct path is plotted in red; it peaks to 0.04 rad and decays to zero in 20–30 sampling periods. The blue trace corresponds to the output error obtained with K_P gain located in the feedback path. The steady-state output error is approximately 0.1 rad. The second figure produced by *P7_2cmd* gives the output position. With the K_P gain in the direct path (red), the model overshoots the target.

S7.3

The overshoot in the step response is associated with conjugate complex poles, resulting in damped oscillations. However, the presence of real zeros may contribute to an overshoot, even in cases when the closed-loop poles are real. The sample transfer function $W_{SS}(s) = (s+1)/(s+4)/(s+2.5) = (s+1)/(s^2 + 6.5s + 10)$ has two real poles and one zero, and it provides the step response with an overshoot:

```
>> num = [ 1 1]
>> den = [0.1 0.65 1]
>> step(num,den)
```

Similarly, the PID position controller with K_P gain in the direct path adds the closed- loop zero $z = p/(p+i)$ to the closed-loop system transfer function (Eq. 7.3), thus resulting in an overshoot.

S7.4

The required modifications of the Simulink model are entered in *P7_4.mdl*. The Matlab command file initializes the parameters, invokes the simulation, and plots the results. The ramp profile can be adjusted by the parameter *del*. Use the following commands and observe the overshoot in the output position:

```
>> del = 0; p7_4cmd
>> del = 0.01; p7_4cmd
.....
>> del = 0.07; p7_4cmd
```

S7.5
With ff = KP/2, the tracking error is reduced to zero.

S7.6
The reference time and data marks are entered in vectors *stim* and *sdat*:

```
>> stim = [ 0 0.05 0.1 0.15 0.2 0.25 0.3]
>> sdat = [ 0 0.002 0.006 0.011 0.014 0.016 0.017]
```

In order to obtain the time resolution of 1 ms, another vector is generated:

```
>> stim1 = [0:300]/1000;
```

Linear interpolation is performed by entering the following command:

```
>> sdat1 = interp1(stim,sdat,stim1,'linear')
```

The resulting trajectory is compared to the original by plotting the waveforms:

```
>> stairs(stim,sdat); hold on; stairs(stim1,sdat1,'r')
```

The first and second derivatives are plotted by entering

```
>> figure; plot(diff(sdat1)); figure; plot(diff(diff(sdat1)))
```

With linear interpolation, the first derivative changes in a stepwise manner, while the second derivative contains impulses.

S7.7

The Matlab command file *P7_7cmd.m* initializes the simulation, invokes the model file, and plots the traces. Linear interpolation results in a significant reduction of the driving torque pulses and gives a smooth change of the output position.

S7.8
The Matlab command file *P7_8cmd.m* initializes the simulation, provides for linear and cubic spline interpolation, invokes the model file, and plots the traces. Compared with the traces obtained with linear interpolation, the cubic spline interpolation provides significant reduction in torque pulsations.

S7.9
The desired traces are obtained with

```
>> figure; plot(diff(sdat1));
>> figure; plot(diff(diff(sdat1)));
>> figure; plot(diff(diff(diff(sdat1))));
```

The third derivative changes in a stepwise manner.

S8.1
The resonant and antiresonant frequencies and their damping are calculated as

$$\omega_p = \sqrt{\frac{K_K(J + J)}{J J}} = 1000\frac{\text{rad}}{\text{s}} = 159.15\,\text{Hz}$$

$$\omega_z = \sqrt{\frac{K_K}{J}} = 707\frac{\text{rad}}{\text{s}} = 112.5\,\text{Hz}$$

$$\zeta_p = \sqrt{\frac{K_V^2(J + J)}{4K_K J\ J}} = 0.01$$

$$\zeta_z = \sqrt{\frac{K_V^2}{4K_K J}} = 0.0071$$

S8.2
The desired transfer function is found from Eq. 8.4:

$$W_P(s) = \frac{1}{2Js} \frac{1 + \frac{K_V}{K_K} s + \frac{J}{K_K} s^2}{1 + \frac{K_V}{K_K} s + \frac{J}{2K_K} s^2}$$

The output speed transient response is obtained by entering the following sequence of commands at the Matlab prompt:

```
>> j = 0.001; kk = 500; kv = 0.01;
>> num = [j/kk   kv/kk   1];
>> den = [  j*j/kk   2*j*kv/kk    2*j   0];
>> impulse(num,den,0.3)
```

An increase in the K_V parameter contributes to the oscillation damping. Note that the traces obtained from Matlab exhibit the effects caused by an extremely low damping of the conjugate complex pole and a finite internal sample time used within Matlab for a discrete approximation to the continuous system. In order to improve the precision, the internal step time can be forced to a lower value by entering

```
>> impulse(num,den,0:0.000001:0.3)
```

S8.3

In accordance with Eq. 8.5, the resonant frequency is calculated as f_{TR} = 600 Hz. The traces obtained with optimized gains are generated by

```
>> clear all
>> p8_3cmd
```

The speed and torque responses are unstable. The speed controller feeds the torsional oscillations back into the system and amplifies their amplitude. In order to reach stability, it is necessary to reduce the feedback gains. The Matlab command line

```
>> KP = KP/2;  KI=KI/4;  p8_3cmd
```

provides the speed and torque responses with reduced gains. In order to reach stability, the gains have to be reduced again:

```
>> KP = KP/2;  KI=KI/4;  p8_cmd,
```

reaching $K_P = K_{POPT}/4$ and $K_I = K_{IOPT}/16$. Some oscillations are noted in the speed and torque traces, yet the response is considered acceptable.

S8.4

The rise time of the output speed is found to be $\tau_R \approx 12T = 3.6$ ms. The closed-loop bandwidth estimate is $f_{BW} = 0.3/0.0036\text{s} = 83.3$ Hz. The ratio f_{TR}/f_{BW} is found as 7.2, in accordance with the conclusions given in Section 9.2, stating that the resonant phenomena can be neglected in cases where the ratio f_{TR}/f_{BW} exceeds 7.

S8.5.

The discrete-time speed-controlled system results in the characteristic polynomial of the third order. Therefore, the impact of the proportional and integral gains on the closed-loop bandwidth and damping is not obvious. On the other hand, the *s*-domain simplified representation of the speed-controlled system with the PI controller, given in Fig. 2.2 and Eq. 2.38, relates the integral gain to the undamped natural frequency of the closed-loop poles, and their damping coefficient to the gain K_P:

$$f(s) = s^2 + \frac{K_P + B}{J}s + \frac{K_I}{J} = s^2 + 2\xi\omega_n s + \omega_n^2$$

$$\Rightarrow \omega_n = \sqrt{\frac{K_I}{J}}, \quad \xi = \frac{K_P + B}{2J\omega_n}$$

The closed-loop bandwidth ω_{BW} is proportional to the natural frequency ω_n. Therefore, in cases when the closed-loop bandwidth has to be altered while keeping the damping factor ξ and the response character unaltered, the gains are to be changed as $K_I \sim \omega_{BW}^2$ and $K_P \sim \omega_{BW}$ (namely, $K_I/K_P \sim \omega_{BW}$).

S8.6

The conjugate complex zeros within the open-loop transfer functions do not contribute to oscillations, even in cases when the relevant damping is extremely low. The transfer function

$$\frac{1}{Js}W_{RR2}(s) = \frac{1}{Js}\frac{1 + \frac{2\zeta_z}{\omega_z}s + \frac{1}{\omega_z^2}s^2}{1 + \frac{2\zeta_z}{\omega_z}s} = \frac{1 + 0.01s + s^2}{s + 0.01s^2}$$

and the relevant step response are obtained by entering

```
>> num = [ 1 0.01 1];  den = [ 0.01   1   0];
>> step(num,den)
```


OG4DPP S. N. Vukosavic boban@ieee.org vukosavic.etf.bg.ac.yu

S8.7

The problem is solved by entering the following sequence of commands:

```
>> x = normrnd(0,1,1000,1);                % Providing the noise signal
>> xs = abs(fft(x));                       % The spectrum of the input to the filter
>> plot(log(xs(1:500)));                   % Verify the uniform distribution
>> numc = [ 1 0.001 1];                    % Notch filter zeros
>> denc = [ 1 1 1 ];                       % Notch filter poles
>> sysc = tf(numc,denc);                   % Continuous-time system
>> sysd = c2d(sysc,1);                     % Discrete domain equivalent
>> [numd,dend] = tfdata(sysd,'v');         % Numerator and denominator of d.t.  t.f.
>> y = filter(numd,dend,x);                % The output of the filter
>> ys = abs(fft(y)); plot(log(ys(1:500)))  % The output spectrum
```

S8.8

The notch-filter zeros are designed to cancel the poles of the mechanical resonator. Therefore, their frequency and damping must correspond to the values obtained from Eq. 8.5. Hence, the notch-filter poles and zeros in continuous time are defined as $\omega_{NF} = \omega_p$, $\xi_P^{NF} = 0.2$, $\xi_Z^{NF} = \xi_p$. The filter design is contained within the Matlab command file *P8_8cmd.m*. The transfer function of discrete-time implementation is given as

$$W_{NF}(z) = \frac{K_3 z^2 - K_4 z + K_5}{z^2 - K_1 z + K_2} \ .$$

The difference equation of the filter is

$$y_{n+1} = K_1 y_n - K_2 y_{n-1} + K_3 x_{n+1} - K_4 x_n + K_5 x_{n-1} \, .$$

The filter coefficients are obtained as

$$K_1 = 1.951, \quad K_2 = 0.9608, \quad K_3 = 0.9812, \quad K_4 = 1.9506, \quad K_5 = 0.9792 \, .$$

S8.9

The problem is solved by using the following Matlab commands:

```
>> numd = [ 1   2   3   2   1 ];  dend = [ 1   0   0   0   0 ];
>> x = [ 1:20];
>> x = x-x;                    % Define the input x as x(k) = 0, except for x(5) = 1
>> x(5) = 1;
>> y = filter(numd,dend,x);
>> stairs(y); axis([0 20 -1 4]);
```