S3.1

The *double ratios* design rule consists of setting $b_{n+2} = b_{n+1}^2/2/b_n$.

$b2 = 0.5,\ \ b3 = 0.125,\ b4 = 0.0156$

The zeros of $f(s)$ are obtained by the Matlab command *roots*([b_4 b_3 b_2 b_1 b_0]) as $s_{1/2} = -2 \pm j2,\ \ s_{3/4} = -2 \pm j2$.

S3.2

(a) The polynomial coefficients are multiplied by 10 with respect to P4.1, while the zeros are unaltered.

(b) $b2 = 500,\ \ b3 = 1250,\ b4 = 1562$
$s_{1/2} = -0.2 \pm j0.2,\ \ s_{3/4} = -0.2 \pm j0.2$ (rad/s)

S3.3

From Eq. 3.19, $K_P = J/2/\tau_{TA} = 5$ kgm^2/s = 5 Nm/(rad/s). The characteristic polynomial is $f(s) = s^2 + s/\tau_{TA} + 1/2/\tau_{TA}^2 = s^2 + 100s + 5000$.
The closed loop poles are $s_{1/2} = -50 \pm j50$ (rad/s).

S3.4

The closed-loop transfer function of the system is obtained in Eq. 3.18:

$$W_{SS}(s) = \frac{K_P}{J\tau_{TA}s^2 + Js + K_P}.$$

The step response is obtained by entering

```
>> j = 0.1; tta = 0.01; kp  = j/2/tta;
>> num = [ kp]; den = [j*tta  j   kp];
>> step(num,den)
```

The overshoot is approximately 4.3%, and the rise time is $\tau_R \approx$ 32 ms.

S3.5

From Eq. 3.27, $K_P = J/2/\tau_{TA} = 5$ and $K_I = B/2/\tau_{TA} = 0.5$. The characteristic polynomial is given in Eq. 3.26. The closed-loop poles are calculated as polynomial zeros:

$s_{1/2} = -50 \pm j50$ (rad/s).

S3.6
The closed-loop transfer function of the system is obtained in Eq. 3.26:

$$W_{SS}(s)=\frac{K_I}{K_I+sB+s^2B\tau_{TA}}\ .$$

The step response is obtained by entering

```
>> b = 0.01; j = 0.1; tta = 0.01; ki  = b/2/tta;
>> num = [ ki]; den = [b*tta  b   ki];
>> step(num,den)
```

The overshoot is approximately 4.3%, and the rise time is $\tau_R \approx$ 32 ms.

S3.7
From Eq. 3.32, $K_P = J/2/\tau_{TA} = 5$ and $K_I = J/8/\tau_{TA}^2 = 125$.
The characteristic polynomial is given in Eq. 3.31. The closed-loop poles are calculated as polynomial zeros:

$s_{1/2}$ = -25 ± j43, s_3 = -50 (rad/s).

S3.8
From Eq. 3.33, the open-loop transfer function $W_S(s)$ is obtained as

$$W_S^{opt}(s)=\frac{1}{8}\frac{1}{(\tau_{TA}s)^2}\frac{1+4\tau_{TA}s}{1+\tau_{TA}s}.$$

It has two poles at the origin, one pole at $p_1 = -1/\tau_{TA}$ = -100 rad/s, and a single zero at z_1 = -25 rad/s. The Bode plot is obtained by entering the following Matlab commands:

```
>> tta = 0.01;
>> num = [4*tta  1];
>> den = [8*tta*tta*tta  8*tta*tta  0  0 ]
>> bode(num,den)
```

Condition $|W_S(j\omega)| = 1$ is obtained for ω_x = 50 rad/s. Notice in the plot that $\omega_x^2 = p_1 z_1$. Compare the plot to Fig. 3.13.

S3.9
The closed-loop transfer function of the system is obtained in Eq. 3.34:

$$W_{SS}^{opt}(s)=\frac{1+4\tau_{TA}s}{1+4\tau_{TA}s+8\tau_{TA}^2s^2+8\tau_{TA}^3s^3}.$$

The step response is obtained by entering

```
>> tta = 0.01;  num = [ 4*tta 1];
>> den = [8*tta*tta*tta  8*tta*tta  4*tta 1];
>> step(num,den)
```

The overshoot is approximately 43.5%, and the rise time is $\tau_R \approx$ 21 ms.

S4.1

The step response obtained with $W(s)$ is obtained by entering the following statements:

```
>> numc = [1];              % Define the numerator of continuous-time  t.f.
>> denc = [1 1];            % Denominator
>> step(numc,denc,'r');     % Obtain the step response as a red color trace
>> hold on;                 % Hold the plot for further comparison
```

Conversion of the s-domain transfer function into the z-domain is obtained by

```
>> sysd = c2d(tf(numc,denc),1,'zoh');
>> [numd, dend] = tfdata(sysd,'v');
```

The step response of the discrete-time system is obtained by

```
>> dstep(numd,dend,'b');
>> axis([0 10 0 1.5]);
```

Note in the figure that the sample train of the discrete-time response coincides with samples of the step response obtained from the continuous-time system.

S4.2

The desired results are obtained by entering the following Matlab statements:

```
>> numc = [1 1];  denc = [1  1  1];
>> step(numc,denc,'r');  hold on;
>> sysd = c2d(tf(numc,denc),1,'zoh');
>> [numd, dend] = tfdata(sysd,'v');
>> dstep(numd,dend,'b');
>> axis([0 10 0 1.5]);
```

S4.3

The sequence of Matlab commands to be repeated is

```
>>  T = 0.1;                                % Enter the sampling time
>>  numc = [1];  denc = [1  0.5  1];
>>  close all;  step(numc,denc);
>>  sysd = c2d(tf(numc,denc),T,'zoh');  figure;
>>  [numd, dend] = tfdata(sysd,'v');  dstep(numd,dend,'b');
```

With $T > T_{MAX} \approx 3$ s, the step response obtained from the discrete-time system loses its resemblance to the original. Consider the poles $s_{1/2} = -0.25 \pm j\,0.96$ of $W(s)$ and their undamped natural frequency of $\omega_n = 1$ rad/s. Then, it is concluded that the output comprises the frequency components at $f_n = 0.159$ Hz. According to the sampling theorem, the sampling frequency $f_S = 1/T$ must be at least two times larger than the maximum frequency contained within the spectrum of the original signal ($f_S > 2\ f_n$). In other words, the sampling process is proper for band-limited input signals. The bandwidth limit $f_{MAX} = f_S/2$ is also known as the Shannon frequency. With $f_n = 0.159$ Hz, the sampling frequency must be in excess to 0.318 Hz, corresponding to $T_{MIN} \approx 3$ s.

S4.4

The desired response is obtained by entering the following Matlab statements:

```
>> p = 0.2027;  i = 0.03512;
>> num = [2*i   0   0];  den = [1  (p+i-2)   (1+i)  -p];
>> dstep(num,den);
```

The resulting figure shows the optimized step response without an overshoot. With reduction of the proportional gain, the step response obtains an overshoot of 34%:

```
>> p = 0.1;  i = 0.03512;   den = [1  (p+i-2)   (1+i)  -p];  dstep(num,den);
```

A reduction in the integral gain does not produce an overshoot, but it increases the rise time and reduces the bandwidth:

```
>> p = 0.2027;  i = 0.01512;  num = [2*i   0   0];
>> den = [1  (p+i-2)   (1+i)  -p];  dstep(num,den);
```

With reference to Eq. 2.38, the characteristic polynomial of the system in Fig. 2.2 can be expressed as $f(s) = s^2 + 2\xi\omega_n s + \omega_n^2$, where the proportional gain affects the damping coefficient while the integral gain determines the undamped natural frequency. In this light, a reduction in the proportional gain decreases the damping and produces an overshoot. On the other hand,

a decrease in the integral gain improves the damping and reduces the natural frequency, diminishing, at the same time, the closed-loop bandwidth and prolonging the rise time.

S4.5
The desired results are obtained by entering the following Matlab statements:

```
>> p = 0.1;  i = 0.03;  numd = [2*i   0   0];  dend = [1  (p+i-2)   (1+i)  -p];
>> dstep(numd,dend);
```

The figure obtained with *dstep* presents the closed-loop step response of the discrete-time system. In order to obtain the continuous-time-domain equivalent, it is necessary to enter

```
>> sysc = d2c(tf(numd,dend,0.001),'zoh')
>> step(sysc);
>> [numc,denc] = tfdata(sysc,'v');
```

In response, Matlab prints the *s*-domain transfer function $W_{ss}(s)$ with two zeros and three poles, and plots the related step response. A comparison of the two waveforms, shows that the overshoot and character of the response correspond.

S4.6
The closed-loop poles and zeros in the *z*- and *s*-planes are obtained as follows:

```
>> roots(numd)          % Zeros of the pulse transfer function
>> roots(dend)          % Poles of the pulse transfer function
>> roots(numc)          % Zeros of the s-domain transfer function
>> roots(denc)          % Poles of the s-domain transfer function
```

The *z*-domain and *s*-domain poles and zeros are mapped with $z = \exp(sT)$, where T stands for the sampling time $T = 0.001$ s. The mapping is obtained with the following commands:

```
>> exp(0.001.*roots(numc))     %  z-domain equivalent of numc zeros
>> exp(0.001.*roots(denc))     %  z-domain equivalent of denc zeros
>> log(roots(numd))/0.001      % s-domain equivalent of numd zeros (err.)
>> log(roots(dend))/0.001      % s-domain equivalent of dend zeros
```

Note that the closed-loop poles in the *s*-domain correspond to the equivalent poles obtained by log(roots(dend))/0.001. On the other hand,

there is no such correspondence between zeros. With the numerator *numd* zeros $z_1 = z_2 = 0$, equivalent *s*-domain zeros are infinite. Note that the option *zoh* within the Matlab command *d2c* (*discrete* to *continuous*) provides the *s*-domain equivalent W(*s*), with samples of its time response $y(kT)$ coinciding with the output samples $y_{(k)}$ of the original discrete-time system. In order to obtain the *s*-domain transfer function with matched poles and zeros, it is necessary to use the function *d2c* with the option *matched*.

S4.7.
The step responses are compared with

```
>>  numd = [1 -0.001];  dend = [1 0.5 0.8];
>> dstep(numd,dend,'r'); hold on; step(d2c(tf(numd,dend,1),'zoh'),'b'),
```

and with

```
>> close all
>> dstep(numd,dend,'r'); hold on; step(d2c(tf(numd,dend,1),'matched'),'b')
```

In both cases, the step response obtained from the discrete-time system is given in red, while the continuous-time domain equivalent is shown in blue. Zooming in on the individual samples shows that the responses correspond with *zoh* option and disagree with *matched* option.

The closed-loop poles and zeros are compared with

```
>> [numc,denc] = tfdata( d2c(tf(numd,dend,1),'zoh'), 'v')
>> roots(dend), exp(roots(denc)*1)
>> roots(numd), exp(roots(numc)*1)
```

and with

```
>> [numc,denc] = tfdata( d2c(tf(numd,dend,1),'matched'), 'v')
>> roots(dend), exp(roots(denc)*1)
>> roots(numd), exp(roots(numc)*1)
```

In both cases, the closed-loop poles are mapped according to $z = \exp(sT)$. With the *zoh* option, the closed-loop zeros disagree. With the option *matched*, the continuous- and discrete-time zeros are related with $z = \exp(sT)$.