LUND
UNIVERSITY

Department of
AUTOMATIC CONTROL

Real-Time Systems

Exam May 3, 2019, hours: 8.00–13.00

Points and grades

All answers must include a clear motivation and a well-formulated answer. Answers may be given in **English or Swedish**. The total number of points is 25. The maximum number of points is specified for each subproblem.

Accepted aid

The textbooks Real-Time Control Systems and Computer Control: An Overview - Educational Version. Standard mathematical tables and authorized "Real-Time Systems Formula Sheet" plus the formula sheet from Reglerteknik AK. Pocket calculator.

Results

The result of the exam will become accessible through LADOK. The solutions will be available on WWW:
http://www.control.lth.se/course/FRTN01/

Solutions to the exam in Real-Time Systems 190503

These solutions are available on WWW:
http://www.control.lth.se/course/FRTN01/

1. Consider the following discrete-time system

$$x(k+1) = ax(k) + u(k)$$

a. For which values of a can the above system have been obtained from a ZOH-sampling of a first-order continuous-time system? (1 p)

b. Assume that $a = 0$. What is the name for a system with this type of dynamics? (1 p)

Solution

a. For a first-order continuous-time system of the form

$$\dot{x}(t) = a_c x(t) + b_c u(t)$$

the corresponding ZOH-sampled version with sampling period h is given by

$$x(k+1) = \exp(a_c h)x(k+1) + \int_0^h \exp(a_c s)ds b_c s u(k)$$

From this we see that

$$a = \exp(a_c h)$$

which can only hold when $a > 0$.

b. This type of system is called a *deadbeat system* or a *pure delay system*.

2. A fixed-point representation of the first-order digital filter

$$\begin{aligned} x(k+1) &= 0.78x(k) - 1.23y(k) \\ u(k) &= 0.91x(k) \end{aligned}$$

with word length $N = 8$ and 6 fractional bits is shown below:

```
int8_t x = 0, y, u;
y = readInput();
u = (int8_t)(((int16_t)58*x) >> 6);
writeOutput(u);
x = (int8_t)(((int16_t)50*x - (int16_t)79*y) >> 6);
```

The variables y, x, and u are all represented as 8-bit signed integers (i.e. 0 fractional bits).

a. What would the consequence be if instead the following implementation was used?

```
int8_t x = 0, y, u;
y = readInput();
u = (58*x) >> 6;
writeOutput(u);
x = (50*x - 79*y) >> 6;
```

(1 p)

b. What would the consequence be if instead the following implementation was used?

```
int8_t x = 0, y, u;
y = readInput();
u = (int8_t)(((int16_t)58*x) >> 5);
writeOutput(u);
x = (int8_t)(((int16_t)50*x - (int16_t)79*y) >> 6);
```

(1 p)

Solution

a. The casts (`int16_t`) are needed to ensure that the multiplications are carried out using 16-bit wordlength. Since x and y can be in the range $[-128, 127]$, the products ($58 \cdot x$, $50 \cdot x$ and $79 \cdot y$) can easily overflow if only 8-bit wordlength is used, giving completely wrong results.

b. Here we rightshift the control signal one bit too little. The effect of this will be that the control signal will be a factor 2 times larger than it should be.

3.

a. Consider the following task set where the standard notation is used. Assign the priorities *High*, *Medium* and *Low* to the three tasks when fixed-priority scheduling with deadline-monotonic priority assignment is used.

Task	T	C	D
A	6	1	5
B	4	2	4
C	3	1	2

(0.5 p)

b. Decide if the task set is schedulable or not using fixed-priority scheduling with deadline-monotonic priority assignment. (1.5 p)

c. Now we instead assume that Earliest Deadline First scheduling should be used. Decide if the task set is schedulable or not. In order to get any points on the problem you may only use methods and results that are part of this course to derive the result. (2 p)

Solution

a. With deadline monotonic priority assignments the task with the shortest relative deadline should have the highest priority, i.e., Task C should have *High* priority, Task B *Medium* priority, and Task A *Low* priority.

b. The sufficient-only test for deadline monotonic priority assignment gives

$$\sum_{i=1}^{n} \frac{C_i}{D_i} = 0.2 + 0.5 + 0.5 = 1.2 > 3(2^{1/3} - 1) = 0.78$$

Hence, this does not provide any answer. Instead we have to use the exact analysis. From this it follows that $R_C = 1 \leq 2$ (OK)

$$\begin{aligned} R_B^0 &= 2 \\ R_B^1 &= 2 + \left\lceil \frac{2}{3} \right\rceil 1 = 3 \\ R_B^2 &= 2 + \left\lceil \frac{3}{3} \right\rceil 1 = 3 \end{aligned}$$

$R_B = 3 \leq 4$ (OK)

$$\begin{aligned} R_A^0 &= 1 \\ R_A^1 &= 1 + \left\lceil \frac{1}{4} \right\rceil 2 + \left\lceil \frac{1}{3} \right\rceil 1 = 4 \\ R_A^2 &= 1 + \left\lceil \frac{4}{4} \right\rceil 2 + \left\lceil \frac{4}{3} \right\rceil 1 = 5 \\ R_A^3 &= 1 + \left\lceil \frac{5}{4} \right\rceil 2 + \left\lceil \frac{5}{3} \right\rceil 1 = 7 \end{aligned}$$

Although R_A has not converged yet, we know already that it is larger than the deadline 5, i.e., the task set is **not** schedulable using deadline monotonic fixed-priority scheduling.

c. Since the deadlines are smaller than the periods the only way to answer this question that is part of the course is to draw the schedule. The schedule is shown in Figure 1. Since the hyper period of the task set is 12 it is sufficient to draw the schedule up to time 12. From the schedule it can be seen that $R_A = 4 \leq 5$, $R_B = 4 \leq 4$, and $R_C = 2 \leq 2$, i.e., the task set is schedulable.

4. When we teach PID control we propose to discretize the three terms (P, I and D) separately. The main reason for this is that it increases the transparency of the algorithm. However, this is not the only way to do it.

a. Consider the following PI controller:

$$U(s) = (K + \frac{K}{T_I s})E(s)$$

Discretize the transfer function above *as a single transfer function* using the Backward Euler approximation, i.e., the two terms should not be discretized individually. Write the result as a difference equation involving the control signal u and the error e and with $u(k)$ only, on the left hand side of the equality sign. (Hint: Start by writing the transfer function with a single fraction bar.) (1 p)

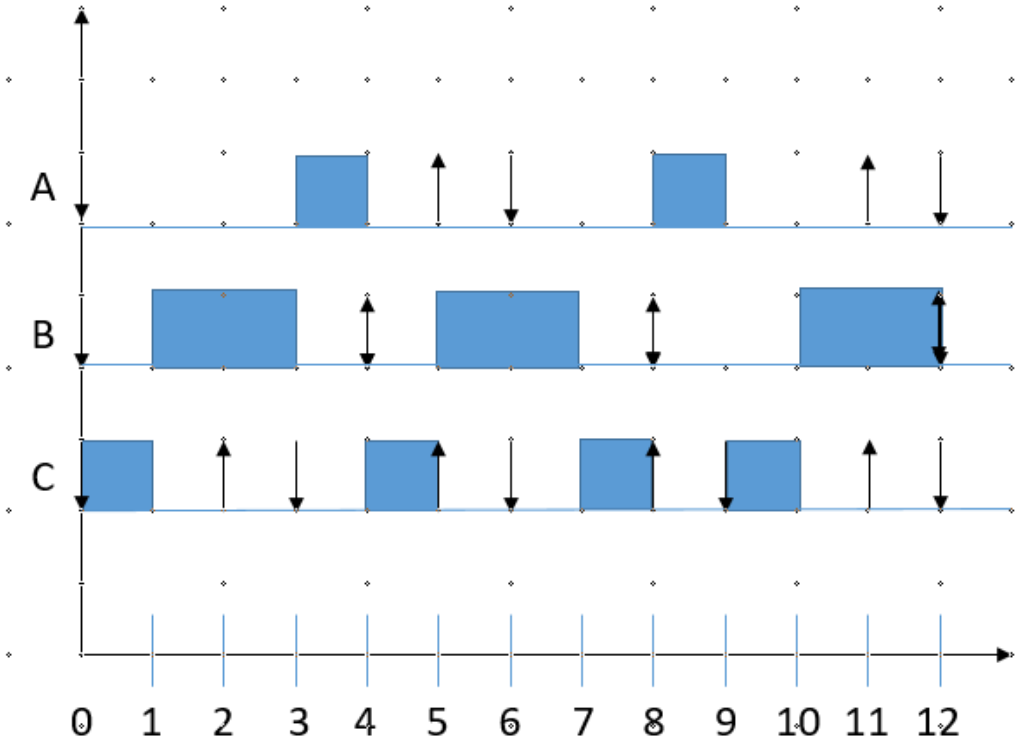

Figure 1 EDF schedule for Problem 3c.

b. Assume now that the actuator used is limited between u_{max} and u_{min}. What unwanted behavior might occur if the above discretized controller is used? (1 p)

c. Modify the controller so that the problem in **b.** is solved. (1 p)

Solution

a.

$$G(s) = K + \frac{K}{T_I s} = \frac{KT_I s + K}{T_I s}$$

This transfer function we now discretize using Backward Euler, i.e., we replace s with $\frac{q-1}{qh}$. Then the resulting difference equation will be

$$u(k) = u(k-1) + (K + Kh/T_I)e(k) - Ke(k-1)$$

b. If the actuator is limited then the control $u(k)$ might wind-up which will eventually cause large over- and undershoots.

c. In order to avoid this the controller should be modified to

$$u(k) = \max(u_{min}, \min(u_{max}, u(k-1) + (K + Kh/T_I)e(k) - Ke(k-1)))$$

5. You are part of a film crew doing a documentary on old propeller planes. At one time, you set up your camera (recording 60 frames per second) and film a plane starting its engine. You watch as the propeller starts to turn with increasing speed, until it reaches a constant angular rate.

Curious, you ask the pilot how fast the propeller was turning when the angular rate was constant, but he only remembers it was somewhere between 2000 and 3000 revolutions per minute (rpm).

a. You watch the film and count the number of revolutions during a short time period. The estimate you get is 1200 rpm. Assuming that the pilot is correct, what is the likely cause for this mismatch? What could have been done differently to ensure that the film would show the correct angular rate of the propeller? (1 p)

b. Assuming that the pilot is correct, what was the exact angular rate of the propeller? (1 p)

Solution

a. The camera records the propeller motion with a sampling frequency of $f_s = 60$ Hz, while the true angular rate of the propeller is between $2000/60 = 33.33$ and $3000/60 = 50$ Hz. Since the rate of the propeller is above the Nyquist frequency ($f_N = f_s/2 = 30$ Hz), the estimate of $1200/60 = 20$ Hz is most likely due to aliasing.

To be completely sure that the correct rate is recorded in the film you would have to use a camera with a faster sampling rate. Only knowing the upper bound 50 Hz, you would need a sampling rate of at least $f_s = 50 \cdot 2 = 100$ Hz.

b. Assuming that the fundamental alias frequency is $f = 1200/60 = 20$ Hz, we have:

$$f = |(f_{\text{true}} + f_N) \text{mod}(f_s) - f_N|.$$

There is only one possible solution for $33.33 \leq f_{\text{true}} \leq 50$ Hz, given by

$$f_{\text{true}} = -f - f_N + f_s + f_N = -f + f_s = -20 + 60 = 40\text{Hz}.$$

Thus the exact angular rate of the propeller was 40 Hz (= 2400 rpm).

6. Consider the following linear system

$$x(k+1) = \begin{pmatrix} 0.5 & 1 \\ 0.5 & 0.7 \end{pmatrix} x(k) + \begin{pmatrix} 0.2 \\ 0.1 \end{pmatrix} u(k).$$

a. Assume state-feedback and design a dead-beat controller for this system. (1 p)

b. Assume that $x(0) = \begin{pmatrix} a & b \end{pmatrix}^\mathsf{T}$. For what values of of a and b will the magnitude of $u(0)$ using the dead-beat controller from **a.** be less than or equal to 10? Draw a figure where the values of a and b that satisfies the inequality are clearly marked out. (1 p)

Solution

a. We are designing a state-feedback controller of the form $u(k) = -Lx(k)$ such that all the closed-loop poles are located in the origin (dead-beat control). Denoting $L = \begin{pmatrix} l_1 & l_2 \end{pmatrix}$ we have the following characteristic polynomial for the closed-loop system:

$$\begin{aligned}\det(zI - (\Phi - \Gamma L)) &= \begin{vmatrix} z + 0.2l_1 - 0.5 & 0.2l_2 - 1 \\ 0.1l_1 - 0.5 & z + 0.1l_2 - 0.7 \end{vmatrix} = \dots \\ &= z^2 + z(0.2l_1 + 0.1l_2 - 1.2) - 0.04l_1 + 0.05l_2 - 0.15\end{aligned}$$

Comparing coefficients with the desired polynomial z^2, we get:

$$\begin{aligned} 0.2l_1 + 0.1l_2 &= 1.2 \\ -0.04l_1 + 0.05l_2 &= 0.15 \end{aligned}$$

with the solution

$$\begin{pmatrix} l_1 & l_2 \end{pmatrix} = \begin{pmatrix} \frac{45}{14} & \frac{39}{7} \end{pmatrix} \approx \begin{pmatrix} 3.21 & 5.57 \end{pmatrix}$$

b. The considered inequality is $|u(0)| \leq 10$, which for the dead-beat control law gives

$$\left|\frac{45}{14}a + \frac{39}{7}b\right| \leq 10.$$

Figure 2 shows the values of a and b that satisfy this inequality. The boundary of the shaded region is given by the two lines $b = -15a/26 + 70/39$ and $b = -15a/26 - 70/39$.

7. For a linear system of the form

$$\begin{aligned} x(k+1) &= \Phi x(k) + \Gamma u(k) \\ y(k) &= Cx(k) \end{aligned}$$

we can design an observer with direct term given by the following system of difference equations:

$$\hat{x}(k) = \Phi\hat{x}(k-1) + \Gamma u(k-1) + K[y(k) - C(\Phi\hat{x}(k-1) + \Gamma u(k-1))]$$

In this problem we will consider the case when Φ, Γ and C are given by

$$\Phi = \begin{pmatrix} 1 & 0.2 \\ 0 & 1 \end{pmatrix}, \quad \Gamma = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad C = \begin{pmatrix} 1 & -1 \end{pmatrix}$$

a. Compute the equations governing the reconstruction error $\tilde{x} = x - \hat{x}$ for the case when $K = \begin{pmatrix} 2 & 4 \end{pmatrix}^\mathsf{T}$. Will the reconstruction error converge to zero?
(1.5 p)

b. Determine K such that the observer poles are placed in $z = 0$ and $z = 0.2$.
(1 p)

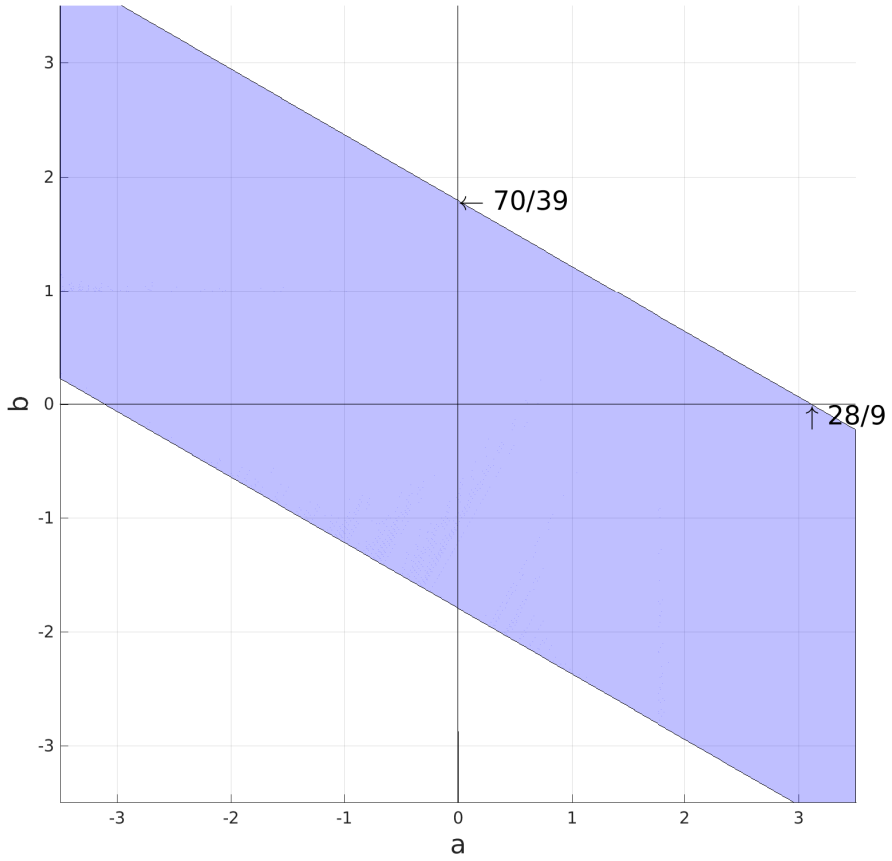

Figure 2 Values of a and b that satisfies the inequality $|u(0)| \leq 10$ (shaded region) in Problem 6b.

c. Let $K = \begin{pmatrix} k_1 & k_2 \end{pmatrix}^\mathsf{T}$. Find a relation between k_1 and k_2 that ensures that the output y of the system is estimated with zero error at all times. (1.5 p)

Solution

a. The reconstruction error $\tilde{x}$ for the observer with direct term is governed by the following system of difference equations:

$$\tilde{x}(k+1) = (I - KC)\Phi\tilde{x}(k) = \left(\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} 2 \\ 4 \end{pmatrix} \begin{pmatrix} 1 & -1 \end{pmatrix} \right) \begin{pmatrix} 1 & 0.2 \\ 0 & 1 \end{pmatrix} \tilde{x}(k)$$
$$= \begin{pmatrix} -1 & 1.8 \\ -4 & 4.2 \end{pmatrix} \tilde{x}(k).$$

The characteristic polynomial of the error dynamics is:

$$\det\left(zI - \begin{pmatrix} -1 & 1.8 \\ -4 & 4.2 \end{pmatrix} \right) = \begin{vmatrix} z+1 & -1.8 \\ 4 & z-4.2 \end{vmatrix} = z^2 - 3.2z + 3$$
$$= (z-1.6)^2 + 3 - 1.6^2 = (z - 1.6 + i\sqrt{0.44})(z - 1.6 - i\sqrt{0.44})$$

We see that the poles are located in $z = 1.6 \pm i\sqrt{0.44}$, which are located outside the unit circle. Therefore the dynamics are unstable, and the reconstruction error will **not** converge to zero.

b. We compute the characteristic polynomial for $K = \begin{pmatrix} k_1 & k_2 \end{pmatrix}^\top$:

$$\det((I - KC)\Phi) = \begin{vmatrix} z + k_1 - 1 & 0.8k_1 - 0.2 \\ k_2 & z - 0.8k_2 - 1 \end{vmatrix} = z^2 + z(k_1 - 0.8k_2 - 2) - k_1 + k_2 + 1.$$

The desired characteristic polynomial is $z(z - 0.2) = z^2 - 0.2z$, which by comparing coefficients gives:

$$\begin{cases} k_1 - 0.8k_2 - 2 = -0.2 \\ -k_1 + k_2 + 1 = 0 \end{cases} \Leftrightarrow \begin{cases} k_1 = 5 \\ k_2 = 4 \end{cases}$$

c. The error $\tilde{y}$ in the estimation of the system output is given by:

$$\begin{aligned} \tilde{y}(k) &= y(k) - C\hat{x}(k) = C\tilde{x}(k) = C(I - KC)\Phi\tilde{x}(k-1) \\ &= (C - CKC)\Phi\tilde{x}(k-1) = (1 - CK)C\Phi\tilde{x}(k-1). \end{aligned}$$

We see that if we choose $CK = 1$, then $\tilde{y} = 0$ at all times. This gives:

$$\begin{aligned} 1 = CK = \begin{pmatrix} 1 & -1 \end{pmatrix} \begin{pmatrix} k_1 \\ k_2 \end{pmatrix} = k_1 - k_2, \\ \Leftrightarrow \\ k_1 = k_2 + 1, \end{aligned}$$

which is the relation between k_1 and k_2 we are looking for.

8. Match the step responses shown in Figure 3 with the pulse transfer functions below:

$$\begin{aligned} H_1(z) &= \frac{0.3}{z - 0.7} \\ H_2(z) &= \frac{1.96}{z^2 + 0.8z + 0.16} \\ H_3(z) &= \frac{0.3}{z^2 - 0.7z} \\ H_4(z) &= \frac{0.34}{z^2 - 1.4z + 0.74} \\ H_5(z) &= \frac{0.3z}{z - 0.7} \\ H_6(z) &= \frac{1.4}{z + 0.4} \end{aligned}$$

Each pulse transfer function corresponds to one step response, and it is assumed in all cases that

$$y(-1) = y(-2) = 0, \quad u(k) = \begin{cases} 0, & k < 0, \\ 1, & k \geq 0. \end{cases}$$

Correct motivations for each matched pair are required for full points. (Hint: Some numerical calculations may be needed to solve this.) (3 p)

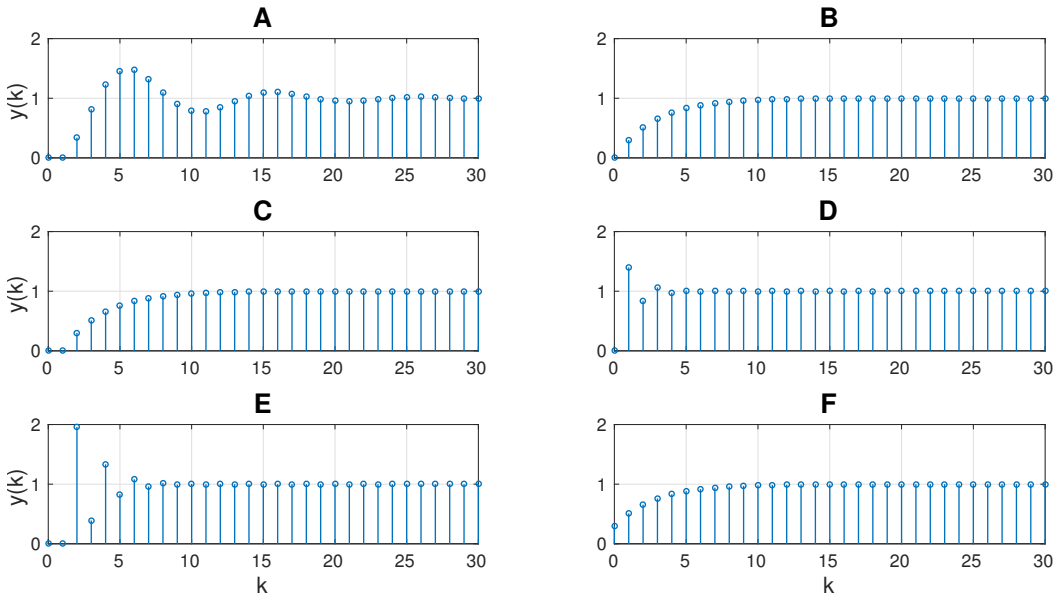

Figure 3 Step responses in Problem 8.

Solution

Starting out, we note in Figure 3 that **B**, **C** and **F** have critically damped step responses, which is only the case when the pulse transfer function have all their poles on the positive (including the origin) real axis within the unit circle. By checking the poles of the pulse transfer functions, we see that H_1, H_3 and H_5 satisfy this. In fact we have

$$H_3(z) = \frac{1}{z}H_1(z), \quad H_5(z) = zH_1(z).$$

This means that the responses of H_3 and H_5 will be identical to those of H_1 with a backward- and forward time-shift respectively. From this analysis we see that

$$\mathbf{B} \Leftrightarrow H_1(z), \quad \mathbf{C} \Leftrightarrow H_3(z), \quad \mathbf{F} \Leftrightarrow H_5(z).$$

Now we have **A**, **D** and **E** left, which should be matched with the two second order systems H_2 and H_4 and the first order system H_6. We note that for **A** and **E** that $y(0) = y(1) = 0$, while for **D** we see $y(1) \neq 0$. Checking the first few time steps for the systems we get

$$\begin{aligned} H_2: \quad & y(k) = -0.8y(k-1) - 0.16y(k-2) + 1.96u(k-2), \\ \Longrightarrow \quad & y(0) = 0,\ y(1) = 0,\ y(2) = 1.96. \end{aligned}$$

$$\begin{aligned} H_4: \quad & y(k) = 1.4y(k-1) - 0.74y(k-2) + 0.34u(k-2), \\ \Longrightarrow \quad & y(0) = 0,\ y(1) = 0,\ y(2) = 0.34. \end{aligned}$$

$$\begin{aligned} H_6: \quad & y(k) = -0.4y(k-1) + 1.4u(k-1), \\ \Longrightarrow \quad & y(0) = 0,\ y(1) = 1.4,\ y(2) = 0.84. \end{aligned}$$

Since $y(1) \neq 0$ only for H_6 we have

$$\mathbf{D} \Leftrightarrow H_6(z).$$

Finally, by comparing the values of $y(2)$ of the two remaining systems and the remaining step responses it is clear that

$$\mathbf{A} \Leftrightarrow H_4(z), \quad \mathbf{E} \Leftrightarrow H_2(z).$$

So in summary:

$$\mathbf{A} \Leftrightarrow H_4(z), \quad \mathbf{B} \Leftrightarrow H_1(z), \quad \mathbf{C} \Leftrightarrow H_3(z), \quad \mathbf{D} \Leftrightarrow H_6(z), \quad \mathbf{E} \Leftrightarrow H_2(z), \quad \mathbf{F} \Leftrightarrow H_5(z).$$

9. Java contains a built-in synchronization mechanism called a *barrier*. Assume that we have a number of threads executing a part of an overall application followed by a point at which the threads must coordinate their results. The barrier is simply a waiting point where all the threads can sync up to either merge results or to safely move on to the next part of the application.

The Java class implementing barriers is called `CyclicBarrier`. The reason it is called *cyclic* is that it can be re-used after the waiting threads have been released.

A slightly simplified version of the interface to `CyclicBarrier` is:

```
public class CyclicBarrier {

public CyclicBarrier(int parties);

public int await();

}
```

The core of the class is the `await()` method. This is called by each thread that needs to wait until the required number of threads are waiting at the barrier. In the constructor the number of threads (parties) using the barrier is specified. This number is used to trigger the barrier; the waiting threads are all released when the number of threads waiting on the barrier is equal to the number of parties. The latter includes the thread that caused the barrier to be triggered.

Each thread that calls the `await()` method gets back a unique return value. This value is related to the arrival order of the thread at the barrier. The first thread that arrives gets a value that is one less than the number of parties and the last thread to arrive will get a value of zero.

The barrier is very simple. All the threads wait until the number of required parties arrive. Upon arrival of the last thread, the waiting threads are released, and the barrier can be re-used. Since the barrier is so simple it is straightforward to implement it using the Java synchronization mechanisms that are part of the course (`synchronized`, `wait()`, `notify()` and `notifyAll()`).

Your task is to implement the class `CyclicBarrier` with the interface and semantics described above. In order to get full points you must ensure that

all threads that are released really will be released, also if some other thread has started to re-use the barrier before the released threads have executed. Correct handling of exceptions is not required. You may also disregard any spurious wake-up issues. (3 p)

Solution

The solution below does not not handle spurious wake-ups:

```
Public class CyclicBarrier {

  private int parties;
  private int barrierLimit;

  public CyclicBarrier(int parties) {
    this.parties = parties;
    barrierLimit = parties;
  }

  public synchronized int await() throws InterruptedException {
    int threadNum;
    parties = parties - 1;
    threadNum = parties;

    if (parties <= 0) {
      parties = barrierLimit;
      notifyAll();
      return threadNum;
    } else {
      try {
        wait();
      } catch (InterruptedException x) {
        parties++;
        throw(x);
      }
      return threadNum;
    }
  }
}
```