DO-DAT Instruction Manual

Hamin Chang
hmchang@cdsl.kr

Seoul National University — July 6, 2020

Introduction

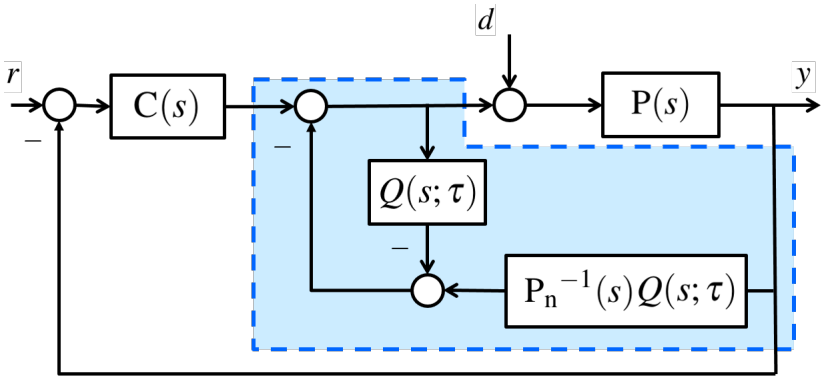

Figure 1: Block diagram of the system with Q-filter-based DOB (sky-blue dashed block).

The standard structure of the Q-filter-based DOB and the closed-loop system are depicted in Figure 1. In the figure, $\mathrm{P}(s)$ and $\mathrm{P_n}(s)$ represent the real plant and its nominal model, respectively, $\mathrm{C}(s)$ is a proper (implementable) controller which is usually designed a priori for $\mathrm{P_n}(s)$, and $Q(s;\tau)$ is a stable low-pass filter called Q-filter which has the form

$$\begin{aligned} Q(s;\tau) &= \frac{c_k(\tau s)^k + c_{k-1}(\tau s)^{k-1} + \cdots + c_0}{(\tau s)^l + a_{l-1}(\tau s)^{l-1} + \cdots + a_1(\tau s) + a_0} \\ &=: \frac{N_Q(s;\tau)}{D_Q(s;\tau)}, \end{aligned}$$

where the real positive τ determines the time constant or the bandwidth and $a_0 = c_0$ for the unity dc gain.

DO-DAT helps you design the Q-filter for robust stability of the closed-loop system against a given variation of uncertain parameters. This article is the instruction manual of DO-DAT. For convenience, it is considered that

- $\mathrm{C}(s) = 2/(s+4)$,
- $\mathrm{P_n}(s) = 5/(s-2)$,
- $\mathrm{P}(s) \in \mathcal{P} := \{\beta_0/(s+\alpha_0) : 4 \le \beta_0 \le 10,\ -10 \le \alpha_0 \le 10\}$

in the rest of this article.

0 Unified function

0.1 DO_DAT.m

```
[Q, Qcanon, supTau] = DO_DAT(sysEnv, 'exact')
[Q, Qcanon, supTau] = DO_DAT(sysEnv, 'approx', res)
[Q, Qcanon, supTau] = DO_DAT(sysEnv, 'approx', res, 'manual', udQcanon)
[Q, Qcanon, supTau] = DO_DAT(sysEnv, 'approx', res, 'r.degreeOfQ', n )
[Q, Qcanon, supTau] = DO_DAT(sysEnv, 'approx', res, 'r.degreeOfQ', n,
                                                    'rhoRoots', LHP roots)
```

This function returns `Q` as a recommended transfer function model $Q(s;\tau)$ that robustly stabilizes the closed-loop system with DOB, `Qcanon` as a transfer function model $Q(s;1)$ with a constant numerator that robustly stabilizes the fast dynamics of the closed-loop system, and `supTau` as the supremum τ^* such that for all $0 < \tau < \tau^*$, the closed-loop system with the DOB designed under $Q(s;1)$ is robustly stable.

- `sysEnv` is expected to be the output of the function `setup_sys` in the page 4.
- For the option `exact`, at least 2018a and Symbolic Math Toolbox are required and the output `supTau` becomes the (almost, in the sense of minor numerical errors) exact value of the supremum τ^*.
- For the option `approx`, the output `supTau` becomes an approximate value of the supremum τ^*. In this case, the resolution `res` must be entered as a positive integer.
- For the option `manual`, the user-defined $Q(s;1)$ `udQcanon` that robustly stabilizes the fast dynamics of the closed-loop system must be entered in the form of a transfer function model.
- For the option `r.degreeOfQ`, the desired relative degree of $Q(s;1)$ `n` must be entered as a positive integer.
- For the option `rhoRoots`, `n`-1 (≥ 1) number of stable roots (that lie on the left-half plane) must be entered in the form of a row vector. Refer to the page 5 for detailed explanation of the option.

Figure 2, 3, and 4 shows the output of the function `DO_DAT` for three different cases.

```
>> [Q, Qcanon, supTau] = DO_DAT(sysEnv, 'exact')

Q =

        1
  -------------
  0.04338 s + 1

Continuous-time transfer function.


Qcanon =

    1
  -----
  s + 1

Continuous-time transfer function.


supTau =

    0.0457
```

Figure 2: The output of the function DO_DAT for the option 'exact'.

```
>> [Q, Qcanon, supTau] = DO_DAT(sysEnv, 'approx', 10, 'manual', tf(0.5, [1, 2, 1, 0.5]))

Q =

                        0.5
  ------------------------------------------------
  1.067e-05 s^3 + 0.0009694 s^2 + 0.02202 s + 0.5

Continuous-time transfer function.


Qcanon =

           0.5
  ---------------------
  s^3 + 2 s^2 + s + 0.5

Continuous-time transfer function.


supTau =

    0.0232
```

Figure 3: The output of the function DO_DAT for the option ‘manual’.

```
>> [Q, Qcanon, supTau] = DO_DAT(sysEnv, 'approx', 10, 'r.degreeOfQ', 4, 'rhoRoots', [-1, -2, -3])

Q =

                                1
  ------------------------------------------------------------
  2.773e-09 s^4 + 2.293e-06 s^3 + 0.0005793 s^2 + 0.04354 s + 1

Continuous-time transfer function.


Qcanon =

                1
  -----------------------------
  s^4 + 6 s^3 + 11 s^2 + 6 s + 1

Continuous-time transfer function.


supTau =

    0.0076
```

Figure 4: The output of the function DO_DAT for the option ‘r.degreeOfQ’ and ‘rhoRoots’.

1 Setup

1.1 setup_sys.m

```
sysEnv = setup_sys(N, D, P_n, C)
```

This function returns a structure variable `sysEnv` that contains the information about the system environment, such as the nominal plant $\mathrm{P_n}(s)$, the controller $\mathrm{C}(s)$, and the set of uncertain plants $\mathcal{P}$, that will be used later in designing the DOB.

- `N` and `D` represent the numerator and denominator of a given set of uncertain plants $\mathcal{P}$, respectively, and they must be entered in the form of a cell that contains both upper and lower bounds of all the coefficients as follows.

```
N = {[4, 10]};
D = {1, [-10, 10]};
```

- The nominal plant `P_n` and the controller `C` must be entered in the form of a transfer function model as follows.

```
P_n = tf(5, [1, -2]);
C = tf(2, [1, 4]);
```

The output `sysEnv` also contains the information about the stability of the nominal closed-loop system and minimum phaseness of a given set of uncertain plants. The field `nominalStab` is 1 if the nominal closed-loop system is stable or 0 otherwise. The field `minPhase` is 1 if a given set of uncertain plants does not contain any non-minimum phase systems or 0 otherwise. Figure 5 shows the output of the function `setup_sys`.

```
>> sysEnv = setup_sys(N, D, P_n, C)

sysEnv =

  다음 필드를 포함한 struct:

          P_n: [1×1 tf]
            C: [1×1 tf]
            N: [2×1 double]
            D: [2×2 double]
  nominalStab: 1
     minPhase: 1
```

Figure 5: The output of the function setup_sys.

2 Design of Coefficients of Q-filter

2.1 gen_Qcanon.m

```
Qcanon = gen_Qcanon(sysEnv, n)
Qcanon = gen_Qcanon(sysEnv, n, 'rhoRoots', LHP roots)
```

This function returns a transfer function model $Q(s;1)$ `Qcanon` with a constant numerator a_0 that robustly stabilizes the fast dynamics of the closed-loop system. In other words, the output of this function

$$Q(s;1) = \frac{N_Q(s;1)}{D_Q(s;1)} = \frac{a_0}{s\rho(s) + a_0}$$

guarantees that the characteristic polynomial of the fast dynamics

$$D_Q(s;1) + \left(\lim_{s\to\infty} \frac{\mathrm{P}(s)}{\mathrm{P_n}(s)} - 1\right) N_Q(s;1) = s\rho(s) + a_0 \lim_{s\to\infty} \frac{\mathrm{P}(s)}{\mathrm{P_n}(s)}$$

is Hurwitz for all $\mathrm{P}(s) \in \mathcal{P}$.

- `sysEnv` is expected to be the output of the function `setup_sys`.
- The desired relative degree of $Q(s;1)$ (i.e., the degree of $D_Q(s;1)$) `n` must be entered as a positive integer.
- For the option `rhoRoots`, `n`-1 (≥ 1) number of stable roots (that lie on the left-half plane) must be entered in the form of a row vector. If the option is not used, $\rho(s)$ is set as $(s+1)^{\mathtt{n}-1}$.

Figure 6 shows the output of the function `gen_Qcanon` for two different cases.

```
>> Qcanon_1 = gen_Qcanon(sysEnv, 3)

Qcanon_1 =

           0.5
  ---------------------
  s^3 + 2 s^2 + s + 0.5

Continuous-time transfer function.

>> Qcanon_2 = gen_Qcanon(sysEnv, 3, 'rhoRoots', [-2, -4])

Qcanon_2 =

            1
  ---------------------
  s^3 + 6 s^2 + 8 s + 1

Continuous-time transfer function.
```

Figure 6: The output of the function gen_Qcanon.

2.2 isFastDynamicsStable.m

```
fastDynamicsStab = isFastDynamicsStable(sysEnv, udQcanon)
```

This function returns a logical output `fastDynamicsStab` that equals to 1 if the fast dynamics of the closed-loop system is robustly stable or 0 otherwise.

- `sysEnv` is expected to be the output of the function `setup_sys`.
- The user-defined $Q(s;1)$ `udQcanon` must be entered in the form of a transfer function model and thus, it can be entered as the output of the function `gen_Qcanon` (Of course, this function will return 1 in that case.).

Figure 7 shows the output of the function `isFastDynamicsStable` for three different cases.

```
>> fastDynamicsStab_1 = isFastDynamicsStable(sysEnv, tf(1, [1, 1]))

fastDynamicsStab_1 =

     1

>> fastDynamicsStab_2 = isFastDynamicsStable(sysEnv, Qcanon_1)

fastDynamicsStab_2 =

     1

>> fastDynamicsStab_3 = isFastDynamicsStable(sysEnv, tf([0.25, 0.25], [1, 4, 6, 4, 9, 0.25]))

fastDynamicsStab_3 =

     0
```

Figure 7: The output of the function isFastDynamicsStable.

3 Determination of Bandwidth of Q-filter

3.1 isValidTau.m

```
validity = isValidTau(sysEnv, udQcanon, tau)
```

This function returns a logical output `validity` that equals to 1 if the closed-loop system with the DOB designed under given $Q(s;1)$ and τ is robustly stable or 0 otherwise.

- `sysEnv` is expected to be the output of the function `setup_sys`.
- The user-defined $Q(s;1)$ `udQcanon` that robustly stabilizes the fast dynamics of the closed-loop system must be entered in the form of a transfer function model.
- `tau` must be entered as a positive real number.

Figure 8 shows the output of the function `isValidTau` for two different cases.

```
>> validity_1 = isValidTau(sysEnv, Qcanon_1, 0.005)

validity_1 =

     1

>> validty_2 = isValidTau(sysEnv, Qcanon_1, 0.05)

validty_2 =

     0
```

Figure 8: The output of the function isValidTau.

3.2 get_supTau.m

```
supTau = get_supTau(sysEnv, udQcanon, 'exact')
supTau = get_supTau(sysEnv, udQcanon, 'approx', res)
```

This function returns the supremum τ^* `supTau` such that for all $0 < \tau < \tau^*$, the closed-loop system with the DOB designed under $Q(s;1)$ is robustly stable.

- `sysEnv` is expected to be the output of the function `setup_sys`.
- The user-defined $Q(s;1)$ `udQcanon` that robustly stabilizes the fast dynamics of the closed-loop system must be entered in the form of a transfer function model.
- For the option `exact`, at least 2018a and Symbolic Math Toolbox are required and the output `supTau` becomes the (almost, in the sense of minor numerical errors) exact value of the supremum τ^*.
- For the option `approx`, the output `supTau` becomes an approximate value of the supremum τ^*. In this case, the resolution `res` must be entered as a positive integer.

Figure 9 shows the output of the function `get_supTau` for each option.

```
>> supTau_exact = get_supTau(sysEnv, Qcanon_1, 'exact')

supTau_exact =

    0.0232

>> supTau_approx = get_supTau(sysEnv, Qcanon_1, 'approx', 10)

supTau_approx =

    0.0232
```

Figure 9: The output of the function get_supTau.

4 Verification

By the function `get_supTau`, it is figured out that, under $Q(s;1) = 0.5/(s^3 + 2s^2 + s + 0.5)$ (`Qcanon_1` in Figure 6), the value of the supremum τ^* such that for all $0 < \tau < \tau^*$ the closed-loop system with DOB is robustly stable, is 0.0232. The result can be verified by `wcgain` function of Robust Control Toolbox that calculates the worst-case peak gain of a given uncertain system. Figure 10 shows that for $\tau = 0.0231$, the worst-case peak gain is bounded but Figure 11 shows that for $\tau = 0.0233$, the worst-case peak gain has infinite lower (or upper) bound.

```
tau_pass =

   0.023100000000000


wcg_rToy_pass =

  다음 필드를 포함한 struct:

          LowerBound: 8.005172555630231e+02
          UpperBound: 8.025560208746389e+02
    CriticalFrequency: 1.891909303499678


wcg_dToy_pass =

  다음 필드를 포함한 struct:

          LowerBound: 1.556941866920235e+02
          UpperBound: 1.560583184885427e+02
    CriticalFrequency: 1.891927756942163
```

Figure 10: The worst-case peak gain of the transfer function of r to y and d to y for $\tau = 0.0231$.

```
tau_fail =

   0.023300000000000


wcg_rToy_fail =

  다음 필드를 포함한 struct:

          LowerBound: 3.934348080140565e+04
          UpperBound: Inf
    CriticalFrequency: 1.893884087557331


wcg_dToy_fail =

  다음 필드를 포함한 struct:

          LowerBound: Inf
          UpperBound: Inf
    CriticalFrequency: 1.893974098579554
```

Figure 11: The worst-case peak gain of the transfer function of r to y and d to y for $\tau = 0.0233$.