(Supplement to Chap 7.)

Robust Control Toolbox™
mixsyn

Provide feedback about this page

H_∞ mixed-sensitivity synthesis method for robust control loopshaping design

Syntax

- `[K,CL,GAM,INFO]=mixsyn(G,W1,W2,W3)`
  `[K,CL,GAM,INFO]=mixsyn(G,W1,W2,W3,KEY1,VALUE1,KEY2,VALUE2,...)`

Description

`[K,CL,GAM,INFO]=mixsyn(G,W1,W2,W3)` computes a controller *K* that minimizes the H_∞ norm of the closed-loop transfer function the weighted mixed sensitivity

- $$T_{y_1u_1} \triangleq \begin{bmatrix} W_1S \\ W_2R \\ W_3T \end{bmatrix}$$

where *S* and *T* are called the *sensitivity* and *complementary sensitivity*, respectively and *S, R* and *T* are given by

- $$S = (I + GK)^{-1}$$
  $$R = K(I + GK)^{-1}$$
  $$T = GK(I + GK)^{-1}$$

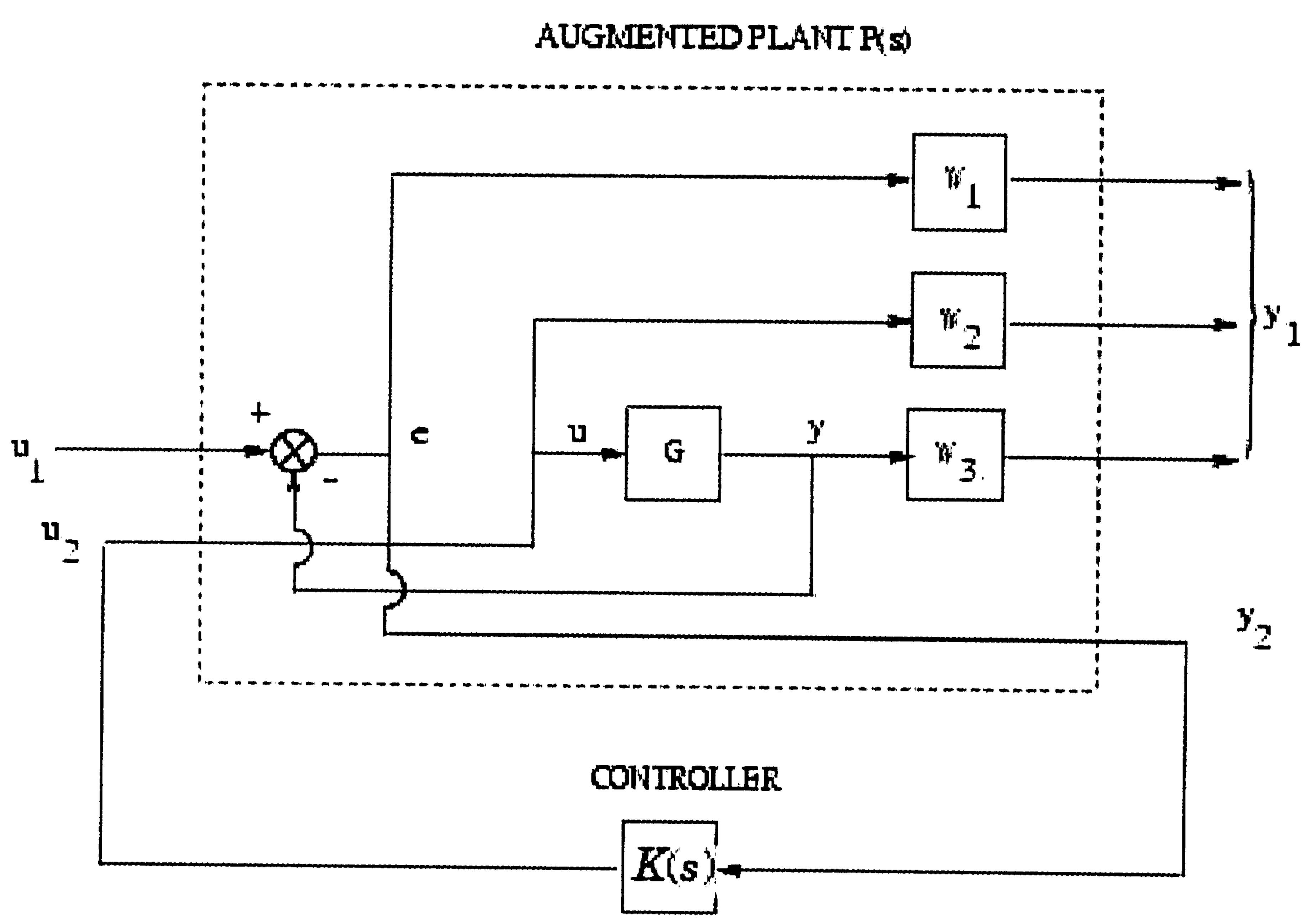


Figure 5-13: Closed-loop transfer function $T_{y_1u_1}$ for mixed sensitivity `mixsyn`.

The returned values of S, R, and T satisfy the following loop shaping inequalities:

$$\bar{\sigma}(S(j\omega)) \le \gamma \; \underline{\sigma}(W_1^{-1}(j\omega))$$

$$\bar{\sigma}(R(j\omega)) \le \gamma \; \underline{\sigma}(W_2^{-1}(j\omega))$$

$$\bar{\sigma}(T(j\omega)) \le \gamma \; \underline{\sigma}(W_3^{-1}(j\omega))$$

where γ =GAM. Thus, W_1, W_3 determine the shapes of sensitivity S and complementary sensitivity T. Typically, you would choose W_1 to be small inside the desired control bandwidth to achieve good disturbance attenuation (i.e., performance), and choose W_3 to be small outside the control bandwidth, which helps to ensure good stability margin (i.e., robustness).

For dimensional compatibility, each of the three weights W_1, W_2 and W_3 must be either empty, scalar (SISO) or have respective input dimensions N_Y, N_U, and N_Y where G is N_Y-by-N_U. If one of the weights is not needed, you may simply assign an empty matrix []; e.g., `P = AUGW(G,W1,[],W3)` is SYS but without the second row (without the row containing `W2`).

Algorithm

```
[K,CL,GAM,INFO]=mixsyn(G,W1,W2,W3,KEY1,VALUE1,KEY2,VALUE2,...)
```

is equivalent to

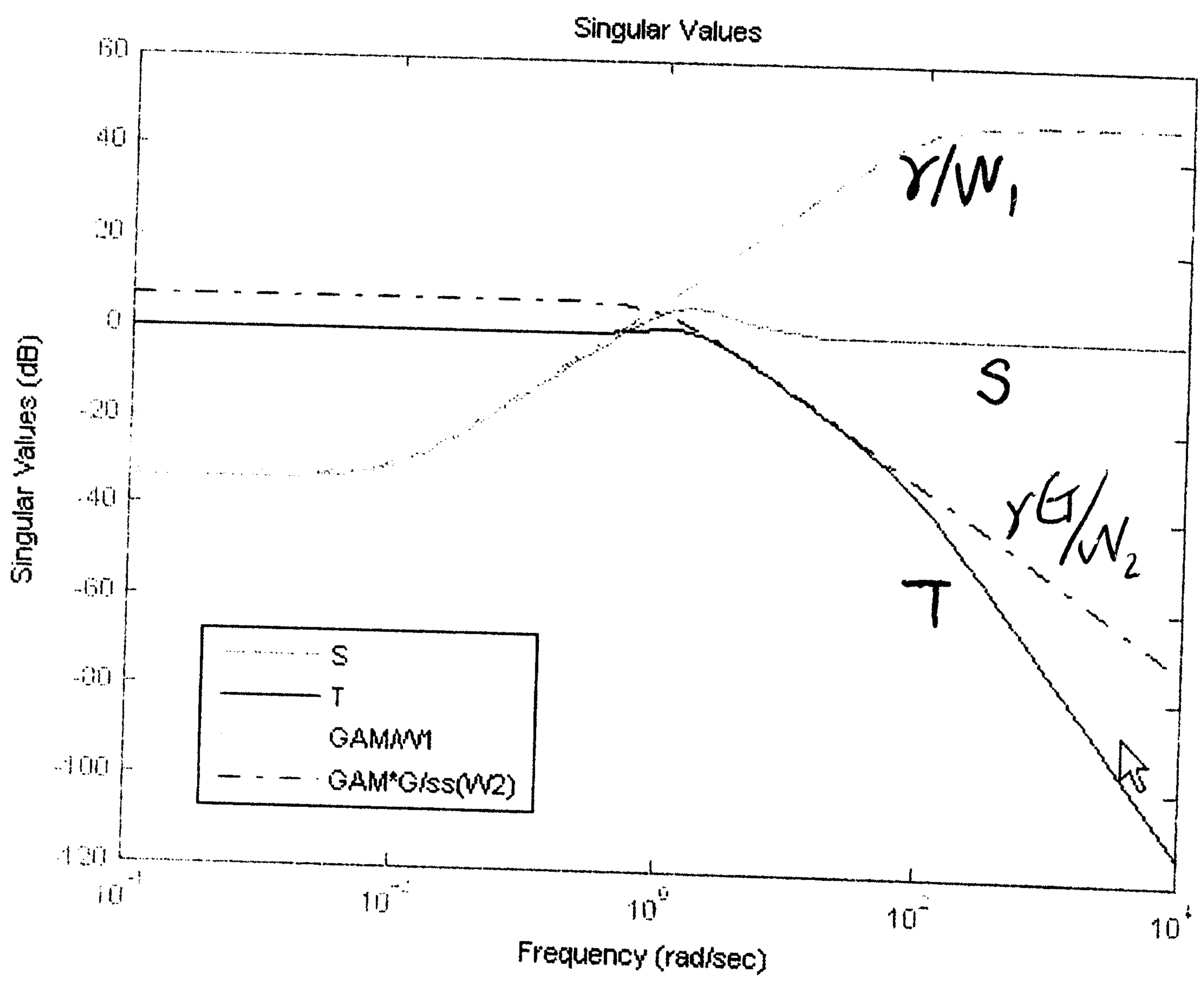
- ```
  [K,CL,GAM,INFO]=...
       hinfsyn(augw(G,W1,W2,W3),KEY1,VALUE1,KEY2,VALUE2,...).
  ```

`mixsyn` accepts all the same key value pairs as `hinfsyn`.

Example

The following code illustrates the use of `mixsyn` for sensitivity and complementary sensitivity `loop-shaping'.

- ```
  s=zpk('s');
  G=(s-1)/(s+1)^2;
  W1=0.1*(s+100)/(100*s+1); W2=0.1;
  [K,CL,GAM]=mixsyn(G,W1,W2,[]);
  L=G*K; S=inv(1+L); T=1-S;
  sigma(S,'g',T,'r',GAM/W1,'g-.',GAM*G/ss(W2),'r-.')
  ```



⇒ $|S| \le \gamma/W_1$

⇒ $R = \frac{K}{1+GK} \le \gamma/W_2$

Figure 5-14: **`mixsyn(G,W1,W2,[ ])` shapes sigma plots of *S* and *T* to conform to γ/W_1 and $\gamma G/W_2$, re**

Limitations

The transfer functions G, W_1, W_2 and W_3 must be proper, i.e., bounded as $s \to \infty$ or, in the discrete-time case, as $z \to \infty$. Additionally, W_1, W_2 and W_3 should be stable. The plant G should be stabilizable and detectable; else, `P` will not be stabilizable by any `K`.

See Also

Augments plant weights for control design

H_∞ controller synthesis