Using Minimal Polynomial Bases for Model-Based Fault Diagnosis – A Demonstration Document for PolyX, Ltd

Erik Frisk and Mattias Nyberg
Vehicular Systems group
Department of Electrical Engineering, Linköping University
SE-581 83 Linköping, Sweden
Email: frisk@isy.liu.se, matny@isy.liu.se

Abstract

This document is a demonstration document, prepared for PolyX Ltd., demonstrating the use of the Polynomial Toolbox for Matlab when designing residual generators for fault diagnosis. A brief introduction to the residual generation problem for fault diagnosis in linear systems is given and a solution based on polynomial methods are outlined. Also, a design example, complete with MATLAB code illustrates how the Polynomial Toolbox can be used in the design of residual generators. For more detailed information on the design method, see e.g. (Frisk & Nyberg 1999b, Nyberg & Frisk 1999).

Contents

1 Introduction

A *model based* diagnosis system commonly consists of a *residual generator* followed by thresholds and some decision logic. The residual generator filters the known signals and generates a signal, the *residual*, that should be small (ideally 0) in the fault-free case and large when a fault is acting on the system. In Figure 1, it is illustrated how the residual generator is connected to the real system. The figure also shows that not only the control signal u influences the system, but also disturbances d and the faults f that we wish to detect. Both disturbances and faults are here modeled as input *signals* to the system. In order *not* to make the residual sensitive to the disturbances d, the disturbances must be *decoupled*. By using several residuals, or a vector-valued residual, where not only

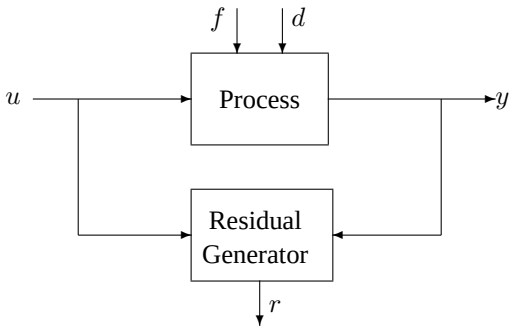

Figure 1: A residual generator.

disturbances but also different subset of faults are decoupled, it is possible to achieve isolation. Isolation means distinguishing between different faults and locate the fault component. This is the basic idea of a diagnosis system using the principle of *structured residuals* (Gertler 1998) or the more general principle of *structured hypothesis tests* (Nyberg 1999). The set of faults that, along with the disturbances, are decoupled in a residual are called *non-monitored* faults.

2 Theory

This demonstration shows *linear* residual generation for *linear* systems with no model uncertainties. A general linear residual generator can be written

$$r = Q(s) \begin{pmatrix} y \\ u \end{pmatrix} \tag{1}$$

i.e. $Q(s)$ is a multi-dimensional transfer-matrix with known signals $y(t)$ and $u(t)$ as inputs and a scalar *residual* as output. The requirement on the residual generator, i.e. $Q(s)$, is that it is sensitive to monitored faults and not sensitive to disturbances (including non-monitored faults).

This section introduces linear residual generation problem and also briefly describes the minimal polynomial basis solution. All derivations are performed in the continuous case but the corresponding results for the time-discrete case can be obtained by substituting s by z and *improper* by *non-causal*.

2.1 Linear Residual Generation

The systems studied in this work are assumed to be on the form

$$y = G_u(s)u + G_d(s)d + G_f(s)f \tag{2}$$

where $y(t)$ is measurements, $u(t)$ is known inputs to the system, $d(t)$ is unknown disturbances including non-monitored faults, and $f(t)$ is the monitored faults. The filter $Q(s)$ in (1) is a residual generator if and only if the transfer function from u and d to r is zero, i.e. $\lim_{t\to\infty} r(t) = 0$ for all $d(t)$ and $u(t)$ when $f(t) = 0$. To be able to detect faults, it is also required that $r(t) \neq 0$ when $f(t) \neq 0$.

Inserting (2) into (1) gives

$$r = Q(s) \begin{bmatrix} G_u(s) & G_d(s) \\ I & 0 \end{bmatrix} \begin{bmatrix} u \\ d \end{bmatrix} + Q(s) \begin{bmatrix} G_f(s) \\ 0 \end{bmatrix} f$$

To make $r(t) = 0$ when $f(t) = 0$, it is required that disturbances and the control signal are *decoupled*, i.e. for $Q(s)$ to be a residual generator, it must hold that

$$Q(s) \begin{bmatrix} G_u(s) & G_d(s) \\ I & 0 \end{bmatrix} = 0 \tag{3}$$

Also, it is required that

$$Q(s) \begin{bmatrix} G_f(s) \\ 0 \end{bmatrix} \neq 0 \tag{4}$$

with suitable properties, e.g. adequate DC-gain from faults. Thus, the linear residual generation problem is to find a suitable $Q(s)$ that fulfills (3) and (4).

2.2 A Solution based on Minimal Polynomial Bases

Equation (3) implies that $Q(s)$ must belong to the left null-space of

$$M(s) = \begin{bmatrix} G_u(s) & G_d(s) \\ I & 0 \end{bmatrix} \tag{5}$$

This null-space is denoted $\mathcal{N}_L(M(s))$. The matrix $Q(s)$ thus need to fulfill two requirements: belong to the left null-space of $M(s)$ *and* give good fault sensitivity properties in the residual. If, in a first step of the design, *all* $Q(s)$ that fulfills the first requirement is found, then a $Q(s)$ with good fault sensitivity properties can be selected. Thus, in a first step of the design of the residual generator $Q(s)$ we need not consider f or $G_f(s)$. The problem is then to find *all* rational $Q(s) \in \mathcal{N}_L(M(s))$. Of special interest are the residual generators with least McMillan degree, i.e. the number of states in a minimal realization.

Finding all $Q(s) \in \mathcal{N}_L(M(s))$ can be done by finding a minimal basis for the rational vector-space $\mathcal{N}_L(M(s))$. A minimal basis for a rational vector-space is a *polynomial* basis (Forney 1975). For now, assume that such a basis can be found and let the base vectors be the rows of a matrix denoted $N_M(s)$. How to extract such a basis will be dealt with in Section 2.3. By inspection of (5), it can be realized that the dimension of $\mathcal{N}_L(M(s))$ (i.e. the number of rows of $N_M(s)$) is

$$\dim \mathcal{N}_L(M(s)) = m - \text{rank } G_d(s) = m - k_d$$

where m is the number of outputs, i.e. the dimension of $y(t)$, and k_d is the number of disturbances, i.e. the dimension of $d(t)$. The last equality holds only if rank $G_d(s) = k_d$, but this should be the normal case.

When a polynomial basis $N_M(s)$ has been obtained, the second and final step in the residual generator design is to shape fault-to-residual responses as described next. The minimal polynomial basis $N_M(s)$ is irreducible and because of this (Kailath 1980), all decoupling residual generators $Q(s)$ can be parameterized as

$$Q(s) = \varphi(s) N_M(s) \tag{6}$$

where $\varphi(s)$ is an arbitrary rational vector of suitable dimensions. This parameterization vector $\varphi(s)$ can e.g. be used to shape the fault-to-residual response or simply to select one row in $N_M(s)$. Since $N_M(s)$ is a basis, the parameterization vector $\varphi(s)$ have minimal number of elements.

The only constraint on $\varphi(s)$ if the residual generator $Q(s)$ is to be realizable, it must be chosen such that (6) is proper. This means that the dynamics, i.e. poles, of the residual generator $Q(s)$ can be chosen freely. This also means that the minimal order of a realization of a decoupling filter is determined by the row-degrees of the *minimal* polynomial basis $N_M(s)$.

2.3 Finding the Minimal Polynomial Basis

The problem of finding a minimal polynomial basis to the left null-space of the rational matrix $M(s)$ can be solved by a transformation to a problem of finding a minimal polynomial basis to the left null space of a polynomial matrix. This transformation can be done in several different ways. In this section, two possibilities are demonstrated, where one is used if the model is given on transfer function form and the other if the model is given in state-space form.

The motivation for this transformation to a purely polynomial problem, is that there exists well established theory (Kailath 1980) for polynomial matrices. In addition, the generally available software, (*The Polynomial Toolbox 2.0 for Matlab 5* 1998), contains an extensive set of tools for numerical handling of polynomial matrices.

State-Space Solution

Assume that the system is described in state-space form,

$$\dot{x}(t) = Ax(t) + B_u u(t) + B_d d(t) \tag{7a}$$
$$y(t) = Cx(t) + D_u u(t) + D_d d(t) \tag{7b}$$

Then it is convienient to use the *system matrix* in state-space form (Rosenbrock 1970) to find the left null-space to $M(s)$. Denote the system matrix $M_s(s)$, describing the system with disturbances as inputs:

$$M_s(s) = \begin{bmatrix} C & D_d \\ -(sI - A) & B_d \end{bmatrix}$$

Also define the matrix P as

$$P = \begin{bmatrix} I & -D_u \\ 0 & -B_u \end{bmatrix}$$

Then the following theorem gives a direct method on how to find a minimal polynomial basis to $\mathcal{N}_L(M(s))$ via the system matrix.

Theorem 1. *Let $V(s)$ be a minimal polynomial basis for $\mathcal{N}_L(M_s(s))$ and let the pair $\{A, [B_u\ B_d]\}$ be controllable. Then $W(s) = V(s)P$ is a minimal polynomial basis for $\mathcal{N}_L(M(s))$.*

The proof of this theorem can be found in (Frisk & Nyberg 1999a). In conclusion, the problem of finding a minimal polynomial basis to a general rational matrix has been transformed into finding a minimal polynomial basis to a specific matrix pencil, in this case the system matrix $M_s(s)$.

Frequency Domain Solution

If the model is given on transfer function form, one way of transforming the rational problem to a polynomial problem is to perform a right MFD on $M(s)$, i.e.

$$M(s) = \widetilde{M}_1(s)\widetilde{D}^{-1}(s) \tag{8}$$

One simple example is

$$M(s) = \widetilde{M}_1(s)d^{-1}(s)$$

where $d(s)$ is the least common multiple of all denominators. By finding a polynomial basis for the left null-space of the *polynomial* matrix $\widetilde{M}_1(s)$, a basis is found also for the left null-space of $M(s)$. No solutions are missed because $\widetilde{D}(s)$ (e.g. $d(s)$) is of full normal rank. Thus the problem of finding a minimal polynomial basis to $\mathcal{N}_L(M(s))$ has been transformed into finding a minimal polynomial basis to $\mathcal{N}_L(\widetilde{M}_1(s))$.

3 Design Example: Aircraft Dynamics

This model, taken from (Maciejowski 1989), represents a linearized model of vertical-plane dynamics of an aircraft. This section includes MATLAB code of central operations. The inputs and outputs of the model are

	Inputs		Outputs
u_1:	spoiler angle [tenth of a degree]	y_1:	relative altitude [m]
u_2:	forward acceleration [ms^{-2}]	y_2:	forward speed [ms^{-1}]
u_3:	elevator angle [degrees]	y_3:	Pitch angle [degrees]

The following Matlab-code defines model equations:

```
%% Define state-space matrices
>> A = [0,      0, 1.1320,      0,-1
     0,-0.0538,-0.1712,      0,0.0705
     0,      0,      0,      1,0
     0, 0.0485,      0,-0.8556,-1.0130
     0,-0.2909,      0, 1.0532,-0.6859];
>> B = [0,0,0
     -0.1200,1,0
      0,0,0
    4.4190,0,-1.6650
    1.5750,0,-0.0732];

>> C = [eye(3) zeros(3,2)];
>> D = zeros(3,3);

%% Form the transfer functions and initialize some constants
>> Gu = ss(A,B,C,D);
>> n = size(A,1);
>> [nmeas,nctrl]  = size(Gu);
```

Suppose the faults of interest are sensor-faults (denoted f_1, f_2, and f_3), and actuator-faults (denoted f_4, f_5, and f_6). Also, assume that the faults are modeled with additive fault models. The total model, including fault models then becomes:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = G(s) \left(\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} + \begin{bmatrix} f_4 \\ f_5 \\ f_6 \end{bmatrix} \right) + \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix}$$

where $G(s) = C(sI - A)^{-1}B + D$. Thus, the transfer function from fault vector f to measurement vector y becomes, $G_{yf}(s) = [I_3 \; G(s)]$. The design example is to design a residual generator $Q(s)$ that decouples faults in the elevator angle actuator, i.e. f_6 is considered a disturbance during the design. The matrix $G_d(s)$ from Equation 2 corresponds to all signals that are to be decoupled, i.e. considered disturbances. In this case, $G_d(s)$ becomes the column in $G_{yf}(s)$ corresponding to f_6, i.e. the third column in $G(s)$. Matrix $G_f(s)$ corresponds to the monitored faults and therefore $G_f(s)$ becomes the rest of the columns of $G_{yf}(s)$. The following Matlab-code defines the transferfunctions from faults and disturbances.

```
>> Gd = Gu(:,3);
>> Gyf = [eye(3,3) Gu];
>> ndist = size(Gd,2);
>> nfault = size(Gyf,2);
```

The next step in the design is to find $N_M(s)$. Before using Theorem 1, the controllability requirement needs to be checked. Matlab gives

```
>> B_d = B(:,3);
>> D_d = D(:,3);
>> rank(ctrb(A,[B B_d]))
```

```
ans =

     5
```

i.e. the controllability requirement is fulfilled. The Polynomial Toolbox macro `null` (computation of the right null-space of a polynomial matrix) can then be used to compute the required left null-basis with the help of the transpose operator

```
>> Ms = [C D_d;-(s*eye(n)-A) B_d)];
>> P = [eye(nmeas) -D;zeros(n,nmeas) -B];
>> Nm = null(Ms.').'*P

Nm =

  0.071s          0.054 + s     0.091                0.12    -1   0
  15s + 23s^2    -6.7          -17 - 0.94s + s^2     31       0   0
```

The second method, the frequency domain approach can also be used. Direct calculations give

```
>> M = minreal([Gu Gd;eye(nctrl) zeros(nctrl,ndist)]);
>> [N,D] = ss2rmf(M.a, M.b, M.c, M.d);
>> Nm2 = null(N.').'

Nm2 =

  0.071s          0.054 + s     0.091                0.12    -1   0
  15s + 23s^2    -6.7          -17 - 0.94s + s^2     31       0   0
```

where `ss2rmf` is a macro of the Polynomial Toolbox that converts a state-space model to a right MFD. As can be seen, the bases are identical. A minimal polynomial basis is not unique, but the algorithm implemented in the toolbox `null` command is based on a canonical echelon form resulting in identical bases. The above basis has a natural interpretation. It means that the following equations hold in the fault-free case and also for all values of the actuator fault $f_6 = d$.

$$0.071\dot{y}_1 + \dot{y}_2 + 0.054y_2 + 0.091y_3 + 0.12u_1 - u_2 = 0 \tag{9}$$

$$23\ddot{y}_1 + 15\dot{y}_1 - 6.7y_2 + \ddot{y}_3 - 0.94\dot{y}_3 - 17y_3 + 31u_1 = 0 \tag{10}$$

The final step in the design is to choose the parametrization vector $\varphi(s)$. In a first design, utilize the first relation (9) and add low-pass dynamics by setting

$$\varphi(s) = \frac{1}{s+1}[1\ 0] \tag{11}$$

Matlab-code to form the residual generator:

```
>> [Qa,Qb,Qc,Qd] = lmf2ss([1,0]*Nm,s+1);
>> Q = ss(Qa,Qb,Qc,Qd);
```

To evaluate the design, both decoupling properties and fault sensitivity need to be analyzed. Figure 2 shows the transfer functions from faults to the residual and the decoupling of disturbances and control signals is evaluated by calculating $\|Q(s)M(s)\|_\infty = -211$ dB. It is clear that the decoupling has suceeded, down to machine precision, and that there is a nonzero gain from all faults besides f_6 which was to be decoupled.

Since the null-space had two basis vectors, we have some freedom in using both these relationships, i.e. (9) and (10), to form the residual generator. By letting

$$\varphi(s) = \frac{1}{(s+1)^2}[1\ -1], \tag{12}$$

i.e. use a linear combination of the two basis vectors and add 2nd order low-pass dynamics to form the residual generator. Here, at least a 2nd order low-pass link was needed since the row-degree of the second basis vector was 2. The residual generator is also normed to have the same DC-gain as the first design to be able to do some comparison. Matlab-code to form the residualgenerator:

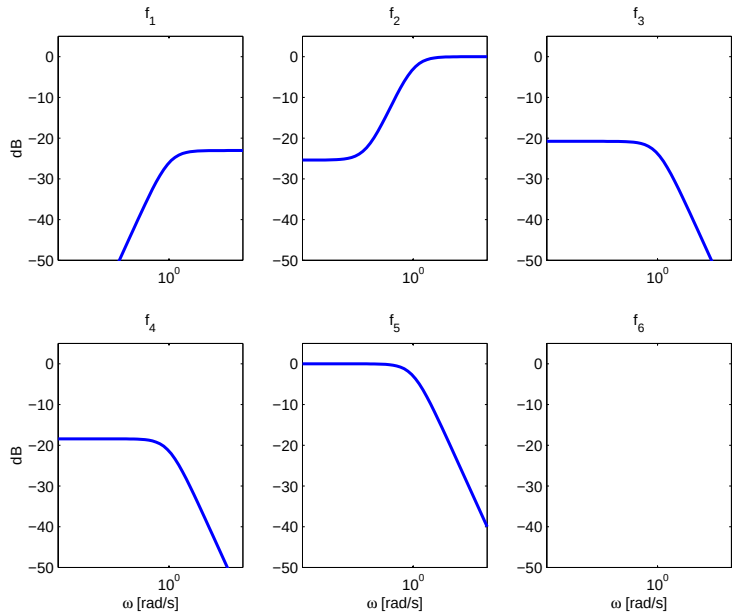

Figure 2: Transfer functions from the 6 faults to the residual. Note that fault 6 is decoupled, i.e. gain 0 from f_6 to r.

```
>> [Q2a,Q2b,Q2c,Q2d] = lmf2ss([1,-1]*Nm,(s+1)^2);
>> Q2 = ss(Q2a,Q2b,Q2c,Q2d);

% Normalize to equal DC-gain of Q(s) and Q2(s)
>> Q2 = svd(evalfr(minreal(Q),0))/svd(evalfr(minreal(Q2),0))*Q2;
```

Evaluation of decoupling and fault sensitivity properties of this second design is shown in Figures 3 where fault sensitivity is shown. calculating $\|Q(s)M(s)\|_\infty = -240$ dB shows that the decoupling, also here has succeded down to machine precision. This second design shows how the design freedom available in $\varphi(s)$ can be used to shape important transfer functions and still keep the important decoupling property.

4 References

Forney, G. (1975). Minimal bases of rational vector spaces, with applications to multivariable linear systems, *SIAM J. Control* **13**(3): 493–520.

Frisk, E. & Nyberg, M. (1999a). A description of the minimal polynomial basis approach to linear residual generation, *Technical Report LiTH-R-2088*, ISY, Linköping, Sweden.

Frisk, E. & Nyberg, M. (1999b). Using minimal polynomial bases for fault diagnosis, European Control Conference, Karlsruhe, Germany.

Gertler, J. (1998). *Fault Detection and Diagnosis in Engineering Systems*, Marcel Dekker.

Kailath, T. (1980). *Linear Systems*, Prentice-Hall.

Maciejowski, J. (1989). *Multivariable Feedback Design*, Addison Wesley.

Nyberg, M. (1999). Framework and method for model based diagnosis with application to an automotive engine, European Control Conference.

Nyberg, M. & Frisk, E. (1999). A minimal polynomial basis solution to residual generation for fault diagnosis in linear systems, Vol. P, IFAC, Beijing, China, pp. 61–66.

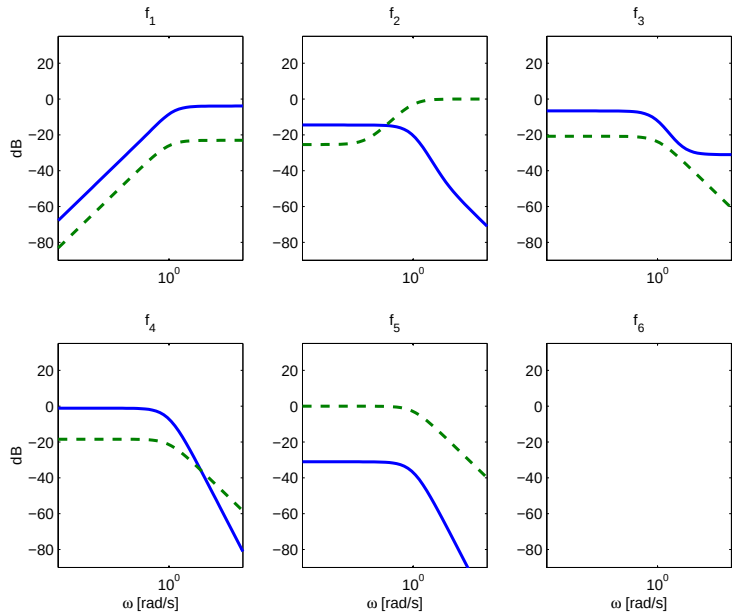

Figure 3: Transfer functions from the 6 faults to the residual. Note that fault 6 is decoupled, i.e. gain 0 from f_6 to r. The dashed lines corresponds to the results of the first design with $\varphi(s)$ given by (11). The solid lines corresponds to the second design where $\varphi(s)$ given by (12).

Patton, R., Frank, P. & Clark, R. (eds) (1989). *Fault diagnosis in Dynamic systems*, Systems and Control Engineering, Prentice Hall.

Rosenbrock, H. (1970). *State-Space and Multivariable Theory*, Wiley, New York.

The Polynomial Toolbox 2.0 for Matlab 5 (1998). Polyx, Czech Republic. URL: http://www.polyx.com.

5 Full Matlab code

In this example, Polynomial Toolbox v2.0 and Control Toolbox v4.2 are used.

```
%% Define state-space matrices
A = [0,      0, 1.1320,      0,-1
     0,-0.0538,-0.1712,      0,0.0705
     0,      0,      0,      1,0
     0, 0.0485,      0,-0.8556,-1.0130
     0,-0.2909,      0, 1.0532,-0.6859];
B = [0,0,0
     -0.1200,1,0
      0,0,0
    4.4190,0,-1.6650
    1.5750,0,-0.0732];

C = [eye(3) zeros(3,2)];
D = zeros(3,3);

%% Form the transfer functions and initialize some constants
Gu = ss(A,B,C,D);
Gd = Gu(:,3);
Gyf = [eye(3,3) Gu];

n = size(A,1);
```

```
[nmeas,nctrl]  = size(Gu);
ndist = size(Gd,2);
nfault = size(Gyf,2);

%% Use state-space solution to find basis N_M(S)
% Check controllability
B_d = B(:,3);
D_d = D(:,3);
rank(ctrb(A,[B B_d]))

% Form system matrix
Ms = [C D_d;-(s*eye(n)-A) B_d];
P = [eye(nmeas) -D;zeros(n,nmeas) -B];
Nm = null(Ms.').'*P

%% Use frequency domain solution to find basis
M = minreal([Gu Gd;eye(nctrl) zeros(nctrl,ndist)]);
[N,D] = ss2rmf(M.a, M.b, M.c, M.d);
Nm2 = null(N.').'

%% Compare the obtained bases
Nm-Nm2


deg(Nm,'row')

%% Design 1
% Select phi(s) = 1/(s+1)[1 0] and form residual generator
[Qa,Qb,Qc,Qd] = lmf2ss([1,0]*Nm,s+1);
Q = ss(Qa,Qb,Qc,Qd);

% Form transferfunctions from faults to residual
Grf = minreal(Q*[Gyf;zeros(nctrl,nfault)]);

% Generate plot to evaluate decoupling properties
sigma(Q*M);

% Generate plot to evaluate fault to residual properties
figure
[gain,phase,freq] = bode(Grf,logspace(-3,2));
gain=shiftdim(gain);
phase=shiftdim(phase);

subplot(231)
semilogx(freq,20*log10(gain(1,:)'));
axis([1e-3 1e2 -50 5])
title('f_1');
ylabel('dB');

subplot(232)
semilogx(freq,20*log10(gain(2,:)'));
axis([1e-3 1e2 -50 5])
title('f_2');

subplot(233)
semilogx(freq,20*log10(gain(3,:)'));
axis([1e-3 1e2 -50 5])
title('f_3');
```

```
subplot(234)
semilogx(freq,20*log10(gain(4,:)'));
axis([1e-3 1e2 -50 5])
title('f_4');
xlabel('\omega [rad/s]');
ylabel('dB');


subplot(235)
semilogx(freq,20*log10(gain(5,:)'));
axis([1e-3 1e2 -50 5])
title('f_5');
xlabel('\omega [rad/s]');

subplot(236)
semilogx(freq,20*log10(gain(6,:)'));
axis([1e-3 1e2 -50 5])
title('f_6');
xlabel('\omega [rad/s]');

%% Design 2
% Select phi(s) = 1/(s+1)^2[1 -1] and form residual generator
[Q2a,Q2b,Q2c,Q2d] = lmf2ss([1,-1]*Nm,(s+1)^2);
Q2 = ss(Q2a,Q2b,Q2c,Q2d);

% Normalize to equal DC-gain of Q(s) and Q2(s)
Q2 = svd(evalfr(minreal(Q),0))/svd(evalfr(minreal(Q2),0))*Q2;

% Form transferfunctions from faults to residual
Grf2 = Q2*[Gyf;zeros(3,6)];

% Generate plot to evaluate decoupling properties
figure
sigma(Q2*M);

% Generate plot to evaluate fault to residual properties
figure
[gain2,phase2,freq2] = bode(Grf2,logspace(-3,2));
gain2=shiftdim(gain2);
phase2=shiftdim(phase2);

subplot(231)
semilogx(freq,20*log10(gain2(1,:)'), freq,20*log10(gain(1,:)'),'--');
axis([1e-3 1e2 -90 35])
title('f_1');
ylabel('dB');

subplot(232)
semilogx(freq,20*log10(gain2(2,:)'), freq,20*log10(gain(2,:)'),'--');
axis([1e-3 1e2 -90 35])
title('f_2');

subplot(233)
semilogx(freq,20*log10(gain2(3,:)'), freq,20*log10(gain(3,:)'),'--');
axis([1e-3 1e2 -90 35])
title('f_3');

subplot(234)
semilogx(freq,20*log10(gain2(4,:)'), freq,20*log10(gain(4,:)'),'--');
```

```
axis([1e-3 1e2 -90 35])
title('f_4');
xlabel('\omega [rad/s]');
ylabel('dB');

subplot(235)
semilogx(freq,20*log10(gain2(5,:)'), freq,20*log10(gain(5,:)'),'--');
axis([1e-3 1e2 -90 35])
title('f_5');
xlabel('\omega [rad/s]');

subplot(236)
semilogx(freq,20*log10(gain2(6,:)'), freq,20*log10(gain(6,:)'),'--');
axis([1e-3 1e2 -90 35])
title('f_6');
xlabel('\omega [rad/s]');
```