

EE263 homework 2

1. *State equations for a linear mechanical system.* The equations of motion of a lumped mechanical system undergoing small motions can be expressed as

$$M\ddot{q} + D\dot{q} + Kq = f$$

where $q(t) \in \mathbf{R}^k$ is the vector of deflections, M , D , and K are the *mass*, *damping*, and *stiffness* matrices, respectively, and $f(t) \in \mathbf{R}^k$ is the vector of externally applied forces. Assuming M is invertible, write linear system equations for the mechanical system, with state

$$x = \begin{bmatrix} q \\ \dot{q} \end{bmatrix},$$

input $u = f$, and output $y = q$.

Solution. We need to express the output q and the state derivative, \dot{q} and \ddot{q} , as a linear function of the state variables q , \dot{q} and the input f . In other words, we should find matrices A , B , C and D such that

$$\begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} = A \begin{bmatrix} q \\ \dot{q} \end{bmatrix} + Bf, \quad q = C \begin{bmatrix} q \\ \dot{q} \end{bmatrix} + Df.$$

Matrices C and D are easy to find: simply, for the second equation to hold we should have

$$C = [I \ 0], \quad D = 0.$$

A and B are a bit harder to find. We will use the differential equation to express \ddot{q} in terms of q , \dot{q} and f . From the given dynamics equation $M\ddot{q} + D\dot{q} + Kq = f$, and assuming M is invertible, we get

$$\ddot{q} = -M^{-1}Kq - M^{-1}D\dot{q} + M^{-1}f,$$

which expresses \ddot{q} in terms of q , \dot{q} , and f . Now we can write the linear dynamical system equations for the system. In block matrix notation we have

$$\frac{d}{dt} \begin{bmatrix} q \\ \dot{q} \end{bmatrix} = \begin{bmatrix} 0 & I \\ -M^{-1}K & -M^{-1}D \end{bmatrix} \begin{bmatrix} q \\ \dot{q} \end{bmatrix} + \begin{bmatrix} 0 \\ M^{-1} \end{bmatrix} u, \quad y = [I \ 0] \begin{bmatrix} q \\ \dot{q} \end{bmatrix},$$

so the matrices in linear dynamical system description are:

$$A = \begin{bmatrix} 0 & I \\ -M^{-1}K & -M^{-1}D \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ M^{-1} \end{bmatrix}, \quad C = [I \ 0], \quad D = 0.$$

2. *Representing linear functions as matrix multiplication.* Suppose that $f : \mathbf{R}^n \rightarrow \mathbf{R}^m$ is linear. Show that there is a matrix $A \in \mathbf{R}^{m \times n}$ such that for all $x \in \mathbf{R}^n$, $f(x) = Ax$. (Explicitly describe how you get the coefficients A_{ij} from f , and then verify that $f(x) = Ax$ for any $x \in \mathbf{R}^n$.) Is the matrix A that represents f unique? In other words, if $\tilde{A} \in \mathbf{R}^{m \times n}$ is another matrix such that $f(x) = \tilde{A}x$ for all $x \in \mathbf{R}^n$, then do we have $\tilde{A} = A$? Either show that this is so, or give an explicit counterexample.

Solution. Any $x = [x_1 \ x_2 \ \cdots \ x_n]^T \in \mathbf{R}^n$ can be written as

$$x = x_1 e_1 + x_2 e_2 + \cdots + x_n e_n$$

where e_i is the i th unit vector in \mathbf{R}^n . From linearity of $f(\cdot)$ we have

$$f(x) = x_1 f(e_1) + x_2 f(e_2) + \cdots + x_n f(e_n),$$

or in (block) matrix form

$$f(x) = [f(e_1) \ f(e_2) \ \cdots \ f(e_n)]x.$$

Therefore, we simply take

$$A := [f(e_1) \ f(e_2) \ \cdots \ f(e_n)].$$

In other words, we only need the transformations of the unit vectors e_i to form the matrix A . Note that $f(e_i) \in \mathbf{R}^m$ for $i = 1, 2, \dots, n$ so A becomes an $m \times n$ matrix. Suppose the matrix A is not unique and there is another $\tilde{A} \in \mathbf{R}^{m \times n}$ such that $f(x) = \tilde{A}x$. Then $Ax = \tilde{A}x$ or $(A - \tilde{A})x = 0$ for all $x \in \mathbf{R}^n$. When $x = e_i$, $(A - \tilde{A})e_i = 0$ implies that the i th column of $(A - \tilde{A})$ is zero. Repeating this argument for $i = 1, 2, \dots, n$ proves that *all* columns of $(A - \tilde{A})$ are zero and hence $A = \tilde{A}$. Therefore the choice of A is *unique*.

3. *Matrix representation of polynomial differentiation.* We can represent a polynomial of degree $< n$,

$$p(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_1x + a_0,$$

as the vector $[a_0 \ a_1 \ \cdots \ a_{n-1}]^T \in \mathbf{R}^n$. Consider the linear transformation \mathcal{D} that differentiates polynomials, *i.e.*, $\mathcal{D}p = dp/dx$. Find the matrix D that represents \mathcal{D} (*i.e.*, if the coefficients of p are given by a , then the coefficients of dp/dx are given by Da).

Solution. According to problem 2 it suffices to compute the transformation of the unit vectors $e_i \in \mathbf{R}^n$ for $i = 1, 2, \dots, n$ under differentiation. In other words

$$D = [De_1 \ De_2 \ \cdots \ De_n].$$

e_1 corresponds to the polynomial $p_1(x) = 1$, and since $\mathcal{D}p_1 = 0$ we have

$$De_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

e_2 corresponds to $p_2(x) = x$ with $\mathcal{D}p_2 = 1$ and therefore

$$De_2 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = e_1.$$

Similarly for $p_3(x) = x^2$ we get $\mathcal{D}p_3 = 2x$ so

$$De_3 = \begin{bmatrix} 0 \\ 2 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = 2e_2.$$

It is easy to see that, in general, for $i > 1$ we have $De_i = (i - 1)e_{i-1}$. Therefore

$$D = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 2 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 3 & \cdots & 0 \\ \vdots & & & & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & n-1 \\ 0 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix}.$$

4. Consider the (discrete-time) linear dynamical system

$$x(t+1) = A(t)x(t) + B(t)u(t), \quad y(t) = C(t)x(t) + D(t)u(t).$$

Find a matrix G such that

$$\begin{bmatrix} y(0) \\ y(1) \\ \vdots \\ y(N) \end{bmatrix} = G \begin{bmatrix} x(0) \\ u(0) \\ \vdots \\ u(N) \end{bmatrix}.$$

The matrix G shows how the output at $t = 0, \dots, N$ depends on the initial state $x(0)$ and the sequence of inputs $u(0), \dots, u(N)$.

Solution. For $t = 0$, $x(t+1) = A(t)x(t) + B(t)u(t)$ becomes

$$x(1) = A(0)x(0) + B(0)u(0).$$

For $t = 1$

$$\begin{aligned} x(2) &= A(1)x(1) + B(1)u(1) \\ &= A(1)[A(0)x(0) + B(0)u(0)] + B(1)u(1) \\ &= A(1)A(0)x(0) + A(1)B(0)u(0) + B(1)u(1). \end{aligned}$$

For $t = 2$

$$\begin{aligned} x(3) &= A(2)x(2) + B(2)u(2) \\ &= A(2)[A(1)A(0)x(0) + A(1)B(0)u(0) + B(1)u(1)] + B(2)u(2) \\ &= A(2)A(1)A(0)x(0) + A(2)A(1)B(0)u(0) + A(2)B(1)u(1) + B(2)u(2). \end{aligned}$$

Now it is easy to guess the general expression for $x(i)$ in terms of $x(0), u(0), \dots, u(i-1)$ (which can be later proved inductively) as follows

$$\begin{aligned} x(i) &= A(i-1)A(i-2) \cdots A(0)x(0) \\ &+ A(i-1)A(i-2) \cdots A(1)B(0)u(0) \\ &+ A(i-1)A(i-2) \cdots A(2)B(1)u(1) \\ &+ A(i-1)A(i-2) \cdots A(3)B(2)u(2) \\ &\vdots \\ &+ A(i-1)B(i-2)u(i-2) \\ &+ B(i-1)u(i-1). \end{aligned}$$

Given $x(i)$ as above, $y(i)$ is simply found from $y(i) = C(i)x(i) + D(i)u(i)$ in terms of $x(0), u(0), \dots, u(i)$ as

$$\begin{aligned}
 y(i) &= C(i)A(i-1)A(i-2) \cdots A(0)x(0) \\
 &+ C(i)A(i-1)A(i-2) \cdots A(1)B(0)u(0) \\
 &+ C(i)A(i-1)A(i-2) \cdots A(2)B(1)u(1) \\
 &+ C(i)A(i-1)A(i-2) \cdots A(3)B(2)u(2) \\
 &\vdots \\
 &+ C(i)A(i-1)B(i-2)u(i-2) \\
 &+ C(i)B(i-1)u(i-1) \\
 &+ D(i)u(i).
 \end{aligned}$$

Therefore

$$G_{i1} = \begin{cases} C(0) & ; \quad i = 1 \\ C(i-1)A(i-2) \cdots A(0) & ; \quad 2 \leq i \leq N+1 \end{cases}$$

and for $i = 1, 2, \dots, N+1$

$$G_{ij} = \begin{cases} 0 & ; \quad N+2 \geq j > i+1 \\ D(i-1) & ; \quad j = i+1 \\ C(i-1)B(i-2) & ; \quad j = i \\ C(i-1)A(i-2) \cdots A(j-1)B(j-2) & ; \quad 2 \leq j < i. \end{cases}$$

5. Express the following statements in matrix language. You can assume that all matrices mentioned have appropriate dimensions. Here is an example: "Every column of C is a linear combination of the columns of B " can be expressed as " $C = BF$ for some matrix F ".

There can be several answers; one is good enough for us.

- (a) For each i , row i of Z is a linear combination of rows i, \dots, n of Y .
- (b) W is obtained from V by permuting adjacent odd and even columns (*i.e.*, 1 and 2, 3 and 4, ...).
- (c) Each column of P makes an acute angle with each column of Q .
- (d) Each column of P makes an acute angle with the corresponding column of Q .
- (e) The first k columns of A are orthogonal to the remaining columns of A .

Solution.

- (a) $Y = UZ$, where U is upper triangular, *i.e.*, $U_{ij} = 0$ for $i > j$.
 (b) $W = VS$, where S is the odd-even switch matrix, defined as

$$W = \begin{bmatrix} e_2 & e_1 & e_4 & e_3 & \cdots & e_m & e_{m-1} \end{bmatrix}.$$

- (c) All entries of the matrix $P^T Q$ are positive.
 (d) The diagonal entries of the matrix $P^T Q$ are positive.
 (e) $A^T A$ has the block diagonal form

$$A^T A = \begin{bmatrix} B_{11} & 0 \\ 0 & B_{22} \end{bmatrix},$$

where $B_{11} \in \mathbf{R}^{k \times k}$.

6. *Gradient of some common functions.* Recall that the gradient of a differentiable function $f : \mathbf{R}^n \rightarrow \mathbf{R}$, at a point $x \in \mathbf{R}^n$, is defined as the vector

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix},$$

where the partial derivatives are evaluated at the point x . The first order Taylor approximation of f , near x , is given by

$$\hat{f}_{\text{tay}}(z) = f(x) + \nabla f(x)^T (z - x).$$

This function is affine, *i.e.*, a linear function plus a constant. For z near x , the Taylor approximation \hat{f}_{tay} is very near f . Find the gradient of the following functions. Express the gradients using matrix notation.

- (a) $f(x) = a^T x + b$, where $a \in \mathbf{R}^n$, $b \in \mathbf{R}$.
 (b) $f(x) = x^T A x$, for $A \in \mathbf{R}^{n \times n}$.
 (c) $f(x) = x^T A x$, where $A = A^T \in \mathbf{R}^{n \times n}$. (Yes, this is a special case of the previous one.)

Solution.

- (a) Expand $f(x)$ as a sum:

$$\begin{aligned} f(x) &= a^T x + b \\ &= \sum_{i=1}^n a_i x_i + b \end{aligned}$$

The j^{th} component of the gradient is: $\frac{\partial f}{\partial x_j} = a_j$. Therefore, $\nabla f(x) = a$.

- (b) Expand $f(x)$ as a double sum (one for each matrix multiplication):

$$\begin{aligned} f(x) &= x^T A x \\ &= \sum_{i=1}^n \sum_{j=1}^n A_{ij} x_i x_j \end{aligned}$$

Now differentiate with respect to x_k , using the product rule:

$$\begin{aligned}\frac{\partial f}{\partial x_k} &= \sum_{i=1}^n \sum_{j=1}^n A_{ij} \left(\frac{\partial x_i}{\partial x_k} x_j + x_i \frac{\partial x_j}{\partial x_k} \right) \\ &= \sum_{i=1}^n \sum_{j=1}^n A_{ij} (\delta_{ik} x_j + x_i \delta_{jk}) \\ &= \sum_{j=1}^n A_{kj} x_j + \sum_{i=1}^n A_{ik} x_i\end{aligned}$$

where we have used the fact that $\frac{\partial x_i}{\partial x_k} = \delta_{ik} = \begin{cases} 0 & \text{if } i \neq k \\ 1 & \text{if } i = k \end{cases}$. It follows that: $\nabla f(x) = (A + A^T)x$.

(c) From the solution of 6b, clearly $\nabla f(x) = 2Ax$.

Note: These derivations start off from the first version of the definition, the “exact” version (because they are formally sound that way). It is then easy to see how to derive the same things from the second version of the definition, the “approximation” version. It’s very similar.

7. *A simple power control algorithm for a wireless network.* First some background. We consider a network of n transmitter/receiver pairs. Transmitter i transmits at power level p_i (which is positive). The path gain from transmitter j to receiver i is G_{ij} (which are all nonnegative, and G_{ii} are positive). The signal power at receiver i is given by $s_i = G_{ii}p_i$. The noise plus interference power at receiver i is given by

$$q_i = \sigma + \sum_{j \neq i} G_{ij} p_j$$

where $\sigma > 0$ is the self-noise power of the receivers (assumed to be the same for all receivers). The *signal to interference plus noise ratio* (SINR) at receiver i is defined as $S_i = s_i/q_i$. For signal reception to occur, the SINR must exceed some threshold value γ (which is often in the range 3 – 10). Various *power control algorithms* are used to adjust the powers p_i to ensure that $S_i \geq \gamma$ (so that each receiver can receive the signal transmitted by its associated transmitter). In this problem, we consider a simple power control update algorithm. The powers are all updated synchronously at a fixed time interval, denoted by $t = 0, 1, 2, \dots$. Thus the quantities p , q , and S are discrete-time signals, so for example $p_3(5)$ denotes the transmit power of transmitter 3 at time epoch $t = 5$. What we’d like is

$$S_i(t) = s_i(t)/q_i(t) = \alpha\gamma,$$

where $\alpha > 1$ is an SINR safety margin (of, for example, one or two dB). Note that increasing $p_i(t)$ (power of the i th transmitter) increases S_i but decreases all other S_j . A very simple power update algorithm is given by

$$p_i(t+1) = p_i(t)(\alpha\gamma/S_i(t)). \tag{1}$$

This scales the power at the next time step to be the power that would achieve $S_i = \alpha\gamma$, if the interference plus noise term were to stay the same. But unfortunately, changing the transmit powers also changes the interference powers, so it’s not that simple! Finally, we get to the problem.

- (a) Show that the power control algorithm (1) can be expressed as a linear dynamical system with constant input, *i.e.*, in the form

$$p(t+1) = Ap(t) + b,$$

where $A \in \mathbf{R}^{n \times n}$ and $b \in \mathbf{R}^n$ are constant. Describe A and b explicitly in terms of σ, γ, α and the components of G .

- (b) *Matlab simulation.* Use matlab to simulate the power control algorithm (1), starting from various initial (positive) power levels. Use the problem data

$$G = \begin{bmatrix} 1 & .2 & .1 \\ .1 & 2 & .1 \\ .3 & .1 & 3 \end{bmatrix}, \quad \gamma = 3, \quad \alpha = 1.2, \quad \sigma = 0.01.$$

Plot S_i and p as a function of t , and compare it to the target value $\alpha\gamma$. Repeat for $\gamma = 5$. Comment briefly on what you observe. *Comment:* You'll soon understand what you see.

Solution.

- (a) The power update rule for a single transmitter can be found by manipulating the definitions given in the problem.

$$\begin{aligned} p_i(t+1) &= \frac{\alpha\gamma p_i(t)}{S_i(t)} = \frac{\alpha\gamma p_i(t)q_i(t)}{s_i(t)} = \frac{\alpha\gamma p_i(t) \left[\sigma + \sum_{j \neq i} G_{ij} p_j(t) \right]}{G_{ii} p_i(t)} \\ &= \frac{\alpha\gamma \left[\sigma + \sum_{j \neq i} G_{ij} p_j(t) \right]}{G_{ii}} \end{aligned}$$

In matrix form the equations look like this:

$$\underbrace{\begin{bmatrix} p_1(t+1) \\ p_2(t+1) \\ p_3(t+1) \\ \vdots \\ p_n(t+1) \end{bmatrix}}_{p(t+1)} = \underbrace{\begin{bmatrix} 0 & \frac{\alpha\gamma G_{12}}{G_{11}} & \frac{\alpha\gamma G_{13}}{G_{11}} & \cdots & \frac{\alpha\gamma G_{1n}}{G_{11}} \\ \frac{\alpha\gamma G_{21}}{G_{22}} & 0 & \frac{\alpha\gamma G_{23}}{G_{22}} & \cdots & \frac{\alpha\gamma G_{2n}}{G_{22}} \\ \frac{\alpha\gamma G_{31}}{G_{33}} & \frac{\alpha\gamma G_{32}}{G_{33}} & 0 & \cdots & \frac{\alpha\gamma G_{3n}}{G_{33}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\alpha\gamma G_{n1}}{G_{nn}} & \frac{\alpha\gamma G_{n2}}{G_{nn}} & \frac{\alpha\gamma G_{n3}}{G_{nn}} & \cdots & 0 \end{bmatrix}}_A \underbrace{\begin{bmatrix} p_1(t) \\ p_2(t) \\ p_3(t) \\ \vdots \\ p_n(t) \end{bmatrix}}_{p(t)} + \underbrace{\begin{bmatrix} \frac{\alpha\gamma\sigma}{G_{11}} \\ \frac{\alpha\gamma\sigma}{G_{22}} \\ \frac{\alpha\gamma\sigma}{G_{33}} \\ \vdots \\ \frac{\alpha\gamma\sigma}{G_{nn}} \end{bmatrix}}_b.$$

- (b) The following matlab code simulates the system for $\gamma = 3$ and an initial power of 0.1 for each transmitter.

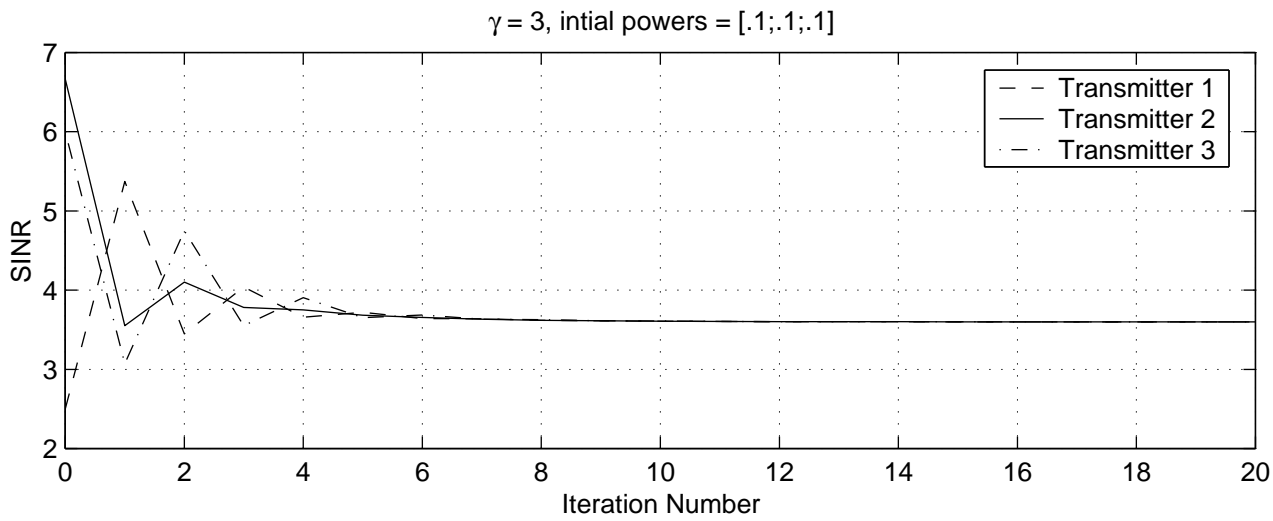
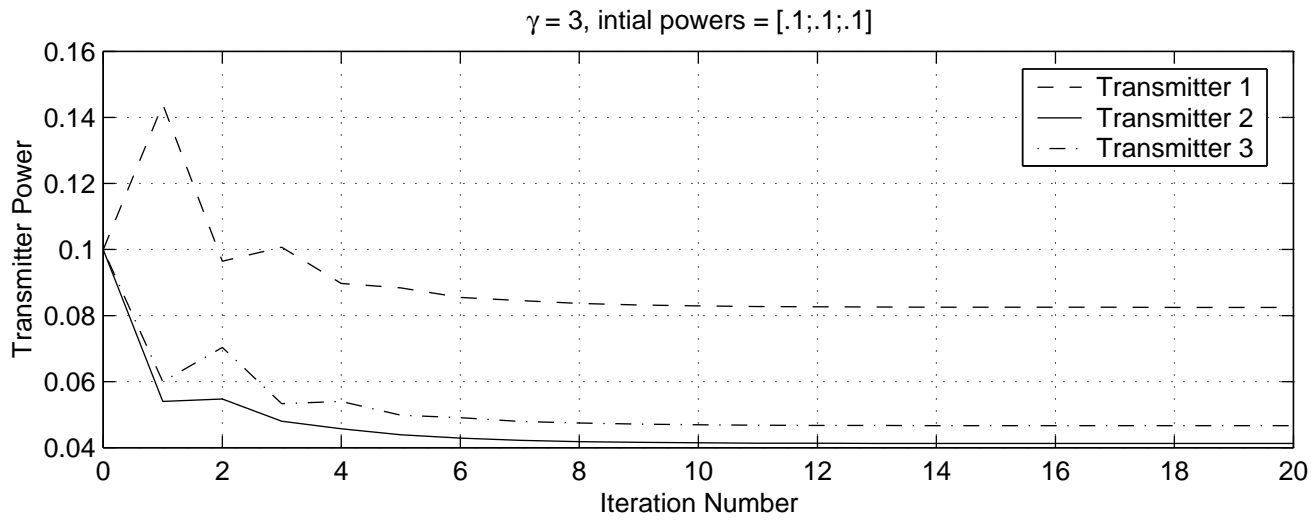
```
clear all; close all;
G = [1 .2 .1; .1 2 .1; .3 .1 3];% Gain matrix
gamma = 3; % minimum SINR
alpha = 1.2; % safety margin
sigma = 0.01; % Noise power (same for all receivers)
A = zeros(3,3); for i = 1:3
for j = 1:3
if (i~=j)
A(i,j) = alpha*gamma*G(i,j)/G(i,i);
end
end
end
b = zeros(3,1); for i = 1:3
b(i) = alpha*gamma*sigma/G(i,i);
end
num_iterations = 20;
p_i = [.1;.1;.1]; % Initialized to p(0)
S = [G(1,1)*p_i(1)/(sigma+G(1,2)*p_i(2)+G(1,3)*p_i(3)); \
G(2,2)*p_i(2)/(sigma+G(2,1)*p_i(1)+G(2,3)*p_i(3)); \
```

```

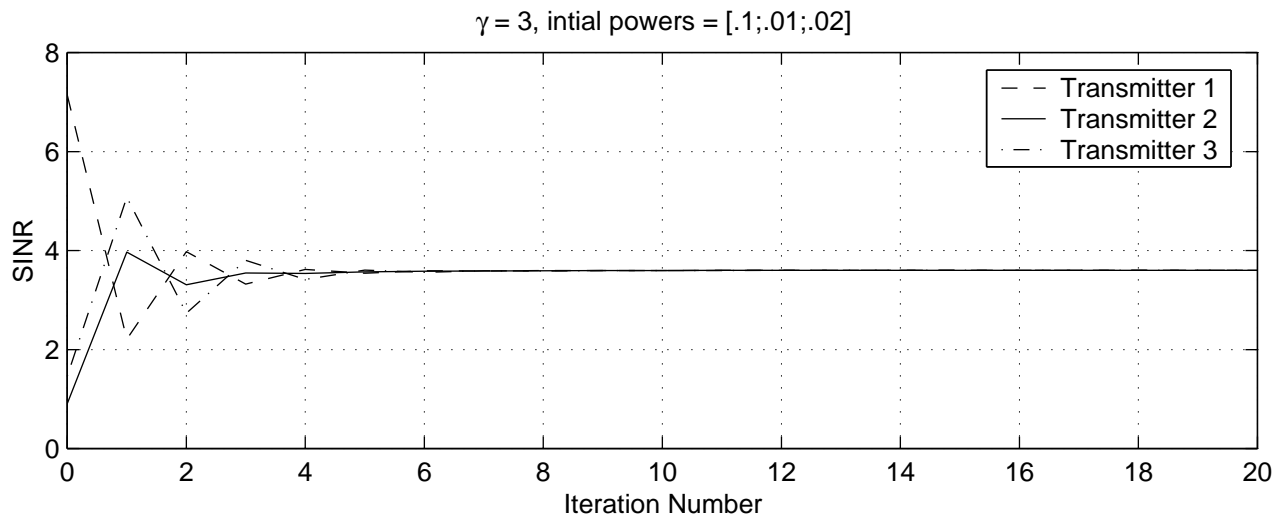
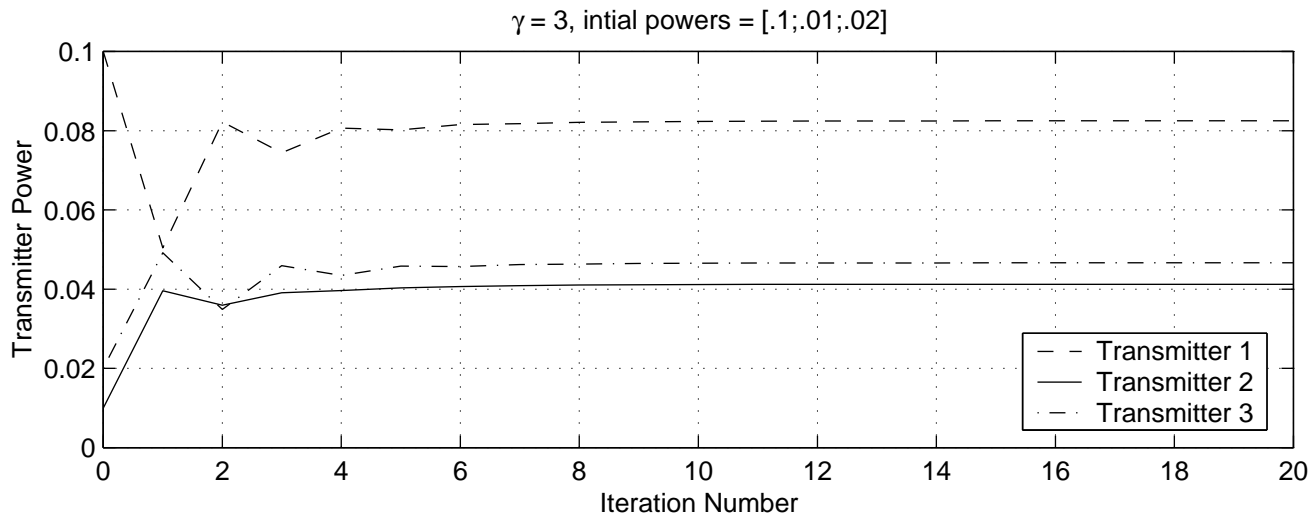
G(3,3)*p_i(3)/(sigma+G(3,1)*p_i(1)+G(3,2)*p_i(2)];
p = p_i; % matrix to store the powers versus time
for i = 1:num_iterations
p_i = A*p_i+b;
p = [p p_i]; % Find the new powers and save
SINR_current = [G(1,1)*p_i(1)/(sigma+G(1,2)*p_i(2)+G(1,3)*p_i(3)); \
G(2,2)*p_i(2)/(sigma+G(2,1)*p_i(1)+G(2,3)*p_i(3)); \
G(3,3)*p_i(3)/(sigma+G(3,1)*p_i(1)+G(3,2)*p_i(2))];
S = [S SINR_current];
end
figure(1); temp = 0:num_iterations; subplot(2,1,1);
plot(temp,p(1,:), '--', temp,p(2,:), '-', temp,p(3,:), '-.');
xlabel('Iteration Number'); ylabel('Transmitter Power');
title('\gamma = 3, intial powers = [.1;.1;.1]');
legend('Transmitter 1', 'Transmitter 2', 'Transmitter 3',0); grid;
subplot(2,1,2); plot(temp,S(1,:), '--',
temp,S(2,:), '-', temp,S(3,:), '-.'); xlabel('Iteration Number');
ylabel('SINR'); title('\gamma = 3, intial powers = [.1;.1;.1]');
legend('Transmitter 1', 'Transmitter 2', 'Transmitter 3',0); grid;

```

The figure below shows the SINR and transmitter power as a function of iteration number.



Similar matlab code can be used to try other initial transmitter powers. For example, the simulation shown below used initial transmitter powers of .1, .01, and .02 for the first, second, and third transmitter respectively. In both cases, the final transmitter powers approach .083, .041, and .047. The SINR approaches $3.6 = \alpha\gamma$. The algorithm appears to work.



Testing the system for $\gamma = 5$ and the same initial conditions (see graphs below) shows that the algorithm does not always succeed. For both initial conditions tried, the transmitter powers grow exponentially. Also, the SINR approaches $5.92 < \alpha\gamma = 6$.

