APPENDIX A

Matlab Code for Data Fitting

A.1 MAMMALIAN HEART RATE PROBLEM

```
File: ls_mammals.m
----------------------Start of actual file contents------------------------
%LEAST SQUARES ANALYSIS OF MAMMALIAN HEART RATE

%w body weights
%r corresponding heart rates
%data is a row vector
w = [3.5 4 6 25 103 117 200 252 300 437 1340 2000 2700 5000
     22500 30000 33000 50000  70000 100000 415000 450000 500000 3000000];
r = [787 660 588 670 347 300 420 352 300 269 251 205 187 120 100
     85 81 70 72 70 45 38 40 48];

x1 = w.^(-1/3) %raise each component of vector w to -1/3 power.
x2 = w.^(-2/3) %the result of this operation is a vector the same size as w
```

Now calculate the slope given by the formula:

$$k = \frac{\sum_{i=1}^{P} r_i w_i^{-1/3}}{\sum_{i=1}^{P} w_i^{-2/3}}$$

We will use the variables numerator and denominator to split up the calculation in the obvious fashion. The numerator is expressed as the vector dot product of r, the row vector of heart rates, and x1 as found above.

```
numerator = r*x1';%apply transpose operator ' to x1 to compute dot product.
denominator = sum(x2);%compute the sum of each component
k1 = numerator/denominator;
```

Now we reproduce the above calculation reproducing all the steps but by using a different data set to compute the slope. It would be more effiecient to pass the data to a subroutine rather than repeat all the code. We examine this in the next section.

```
%%Now build the model on the first 2/3 of the data (16 points)
ws = w(8:24)%the notation 8:24 is equivalent to [8 9 10 11 12 .... 24]
rs = r(8:24)

x1 = ws.^(-1/3)%raise each component of vector w to -1/3 power.
x2 = ws.^(-2/3)

numerator = rs*x1';%apply transpose operator ' to x1 to compute dot product.
denominator = sum(x2);%compute the sum of each component
k2 = numerator/denominator;%see formula in section 3.1.1

hold on
plot(w.^(-1/3),r,'o')%plot raw data
plot(w.^(-1/3),k1*w.^(-1/3),'--x')%plot first model
plot(w.^(-1/3),k2*w.^(-1/3),'--v')%plot second model
title('mammalian heart rate model')
xlabel('weight w^{(-1/3)}')
ylabel('pulse rate')
legend('raw data','least squares fit (all data)', 'least squares fit 2/3 data')
----------------------End of actual file contents--------------------------------
```

A.2 LEAST SQUARES WITH NORMAL EQUATIONS

This program consists of two parts: a subroutine called ls_normal.m and a driver called run_ls.m.

```
File:  ls_normal.m
----------------------Start of actual file contents------------------------
%Input:
%   x is a column vector of domain (input) variables
%   y is a column vector of range (output) variables
%
%Output:
%   m is the slope of the line
%   b is the intercept of the line

function [m,b] = ls_normal (x,y)

P = size(x,1)%how many points are there in this column vector?
```

Now we compute the terms required in the evaluation of m and b in the normal equations. Recall

$$m = \frac{(\sum y_i)(\sum x_i) - P\sum y_i x_i}{(\sum x_i)^2 - P\sum x_i^2}$$

$$b = \frac{-(\sum y_i)(\sum x_i^2) + (\sum x_i)(\sum y_i x_i)}{(\sum x_i)^2 - P\sum x_i^2}$$

We set sy $= \sum y_i$, dp_xy $= \sum y_i x_i$ and x_sq $= \sum x_i^2$.

```
sy = sum(y);%sum y_i (scalar)
sx = sum(x);%sum x_i (scalar)
dp_xy = x'*y; %y dot product x (scalar)
x_sq = sum(x.*x);%sum x_i^2 term (scalar)
denom = sx^2 - P*x_sq; %(scalar)

m = (sy*sx-P*dp_xy)/denom
b = (-sy*x_sq+sx*dp_xy)/denom

%plot results
plot(x,y,'o')
hold
plot(x,m*x+b,'--v')
legend('data','model')
-----------------------End of actual file contents--------------------------------
```

The above subroutine is called using the following driver:

```
File: run_ls.m
----------------------Start of actual file contents------------------------
w = [3.5 4 6 25 103 117 200 252 300 437 1340 2000 2700 5000
     22500 30000 33000 50000 70000 100000 415000 450000 500000 3000000];
r = [787 660 588 670 347 300 420 352 300 269 251 205 187 120 100
     85 81 70 72 70 45 38 40 48];

x = w.^(-1/3)'%note that the transpose operator ' turns the row vec into col vec.
y = r'% the subroutine expects column vectors by design.

[m,b] = ls_normal (x,y)
-----------------------End of actual file contents--------------------------------
```

A.3 LEAST SQUARES WITH OVERDETERMINED SYSTEM

```
File: ls_interp.m
----------------------Start of actual file contents-------------------------
%Input:
%   x is a column vector of domain (input) variables
%   y is a column vector of range (output) variables
%
%Output:
%   m is the slope of the line
%   b is the intercept of the line

function [m,b] = ls_interp(x,y)

%Compute the matrix X
```

Recall the equation we are solving in this problem:

$$\begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_P \end{pmatrix} \begin{pmatrix} b \\ m \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_P \end{pmatrix}$$

In terms of matrices we can summarize the above as

$$\mathrm{Xvec} = \mathrm{y}$$

This equation will be solved via matlab's backslash routine.

```
P = size(x,1)%how big is the data set?

c1 = ones(P,1)%create a column vector of ones of length P

X = [c1 x]% construct the "interpolation matrix"

vec = X\y% solve the least squares problem

b = vec(1)%obtain the first component (intercept)
m = vec(2)%obtain the slope

%plot results
plot(x,y,'o')
```

```
hold
plot(x,m*x+b,'--v')
legend('data','model')
----------------------End of actual file contents--------------------------------
```

A.4 NON-NEWTONIAN FISH

See the discussion in Section 6.5.1.

```
File: fishfit.m
---------------------Start of actual file contents-------------------------
y = log(T(2:101)-T(1:100))%use only the first 100 points
x = log(425-T)%


m11 = x(1:100)*x(1:100)'

m12 = sum(x(1:100))
m21 = m12
m22 = 100
r1 = (x(1:100))*y'
r2 = sum(y(1:100))

R = [r1;r2]

M = [m11 m12; m21 m22]

XX = inv(M)*R%this vector contains the results [b alpha]
k = exp(XX(2))%transforming back to original model equation
----------------------End of actual file contents--------------------------------
```

A.5 PREDITOR OR PREY?

See the discussion in Section 6.5.2.

```
File: predpreyfit.m
---------------------Start of actual file contents-------------------------
a = f(1:19800);%init data representing the population of species A
b = r(1:19800);%init data representing the population of species B

%%FOXES matrix equation coefs
m11 = -sum(a*a')
m12 = sum(a.*a.*b)
m21 = -m12
m22 = sum(a.*a.*b.*b)

z1 = (f(2:19801)-a)*a'
z2 = sum((f(2:19801)-a).*a.*b)
```

```
M = [m11 m12; m21 m22];%now compute the matrix by assembling the components
R1 = [z1;z2]%this is the RHS of equation for (g_1, c_1)

cg1 = inv(M)*R1%solve for (g_1, c_1)


%%rabbits equation coefs
m11 = sum(b*b')
m12 = -sum(b.*b.*a)
m21 = -m12
m22 = -sum(a.*a.*b.*b)

z1 = (r(2:19801)-b)*b'
z2 = sum((r(2:19801)-b).*a.*b)

M = [m11 m12; m21 m22];
R1 = [z1;z2]

cg2 = inv(M)*R1%solve for (g_2, c_2)
----------------------End of actual file contents--------------------------------
```

A.6 TIRE DISTRIBUTOR

See the discussion in Chapter 7 Section 7.1.

```
File: flatdemand.m
---------------------Start of actual file contents-----------------------

function [num_tires] = flatdemand

u = rand(1);
u1 = rand(1);
if u < 12/365
    num_tires = floor(u1*100);
elseif u >= 12/365 & u < 16/365
    num_tires = 100 + floor(u1*200);
elseif u >= 16/365 & u < 43/365
    num_tires = 300 + floor(u1*200);
elseif u >= 43/365 & u < 86/365
    num_tires = 500 + floor(u1*200);
elseif u >= 86/365 & u < 134/365
    num_tires = 700 + floor(u1*200);
elseif u >= 134/365 & u < 201/365
    num_tires = 900 + floor(u1*200);
elseif u >= 201/365 & u < 279/365
    num_tires = 1100 + floor(u1*200);
elseif u >= 279/365 & u < 334/365
    num_tires = 1300 + floor(u1*200);
```

```
elseif u >= 334/365 & u < 356/365
    num_tires = 1500 + floor(u1*200);
elseif u >= 356/365 & u < 363/365
    num_tires = 1700 + floor(u1*200);
else u >= 363/365 & u <1
    num_tires = 1900 + floor(u1*200);
end
----------------------End of actual file contents--------------------------------

File: simulate.m
--------------------Start of actual file contents-------------------------


function [AVE_DAILY_COST] = simulate(num_runs, delivery_interval, delivery_quantity)

NUM_DAYS = 365
TRUCK_CAPACITY=4000;%tires
truck_charge = 400;%delivery cost per truck
delivery_charge = truck_charge*(floor((delivery_quantity-1)/TRUCK_CAPACITY)+1);
penalty = 1000;%10 dollar penalty per day * 1000 tires

for i = 1:num_runs
    day_counter = 0;
    %Assume there is a delivery at the outset
    NUM_TIRES = delivery_quantity;%init
    COST =delivery_charge;
    interest_rate = 0.01;
    days_without_tires =0;
    delivery_counter = 0;

    daily_inventory(1) = NUM_TIRES;

    while day_counter <= NUM_DAYS;
        if delivery_counter == delivery_interval;%add delivery charge
            NUM_TIRES = NUM_TIRES + delivery_quantity;
            COST = COST + delivery_charge;
            delivery_counter = 0;
        end
        NUM_TIRES = NUM_TIRES - flatdemand; %sell tires for day

        if NUM_TIRES <=0;%out of stock?
            days_without_tires = days_without_tires + 1;
            NUM_TIRES = 0;
        end
        COST = COST + interest_rate*NUM_TIRES;%add charges for unsold tires
        day_counter = day_counter + 1;
        daily_inventory(day_counter+1) = NUM_TIRES;
```

```
        delivery_counter = delivery_counter + 1;
    end
    %add daily charge due interest at end of day for remaining tires
    cost_penalty = days_without_tires*10000;
    AVE_DAILY_COST(i) = (COST + cost_penalty)/NUM_DAYS;
end
-----------------------End of actual file contents--------------------------------
```

```
File: run_simulate.m
----------------------Start of actual file contents------------------------

num_runs = 100;
delivery_quantity = 4000; %number to have delivered in each shipment (X in notes)


for j=1:6
    for i = 1:25
    delivery_interval = i;
    DAILY_COST(:,i,j) = simulate(num_runs, delivery_interval, delivery_quantity);
    end
    delivery_quantity = delivery_quantity + 4000
end

for j=1:6
    for i = 1:25
    AVES(i,j) = sum(DAILY_COST(:,i,j))/num_runs
    end
end



semilogy(1:25,AVES(:,1),'.',1:25,AVES(:,2),'x',1:25,AVES(:,3),'o',1:25,AVES(:,4),'+',1:25,
legend('delivery quantity 4000','delivery quantity 8000','delivery quantity 12000','delive
ylabel('average daily cost')
xlabel('delivery interval in days')
-----------------------End of actual file contents--------------------------------
```

A.7 BLACKJACK

```
File: blackjack.m
----------------------Start of actual file contents------------------------
%blackjack simulation
%bet one dollar on each hand; start with $100
my_money = 100;
iwin=0;%counter for number of wins
ilose = 0;%counter for number of losses
ties = 0;%counter for number of ties
hand = 0;%counter for number of hands played.
my_stay_value = 14; %don't take another card if hand is worth 14 or more.

%deal cards from 2 decks
for i = 1:100
%note 11= jack, 12 = queen, 13 = king, and 14 = ace.
d1 = [2 2 2 2 3 3 3 3 4 4 4 4 5 5 5 5 6 6 6 6 7 7 7 7 8 8 8 8 9 9 9 9 10 10 10 10
      11 11 11 11 12 12 12 12 13 13 13 13 14 14 14 14];
```

```
deck = [d1 d1];
numcards = size(deck,2);
bigindexset = 1:numcards;
permuted_indices = bigindexset(randperm(numcards));
shuffled_cards = deck(permuted_indices);


%One game (of two decks)
card_counter = 1;
while card_counter < numcards - 10%dont let deck run out!
   %new hand
   hand = hand +1;
   my_count = 0;
   dealer_count = 0;

   %make initial deal of two cards to Player and Dealer.
   my_count = my_card_value(card_counter, shuffled_cards, my_count);
   card_counter = card_counter+1;%advance deck index
   dealer_count = dealer_card_value(card_counter, shuffled_cards, dealer_count);
   card_counter = card_counter+1;
   my_count = my_card_value(card_counter, shuffled_cards, my_count);
   card_counter = card_counter+1;
   dealer_count = dealer_card_value(card_counter, shuffled_cards, dealer_count);
   card_counter=card_counter+1;

   while my_count < my_stay_value  & card_counter < numcards
      my_count = my_card_value(card_counter, shuffled_cards, my_count);
      card_counter = card_counter +1;
   end

   while dealer_count < 17  & card_counter < numcards & my_count < 22
      dealer_count = dealer_card_value(card_counter, shuffled_cards, dealer_count);
      card_counter = card_counter +1;
   end

%who wins?
   if my_count > 21% I am bust
      my_money = my_money - 1;
      ilose = ilose +1;
   elseif dealer_count > 21%dealer is bust
      my_money = my_money + 1;
      iwin = iwin +1;
   elseif my_count == dealer_count
      %push--my winnings don't change
      ties = ties +1;
   elseif my_count > dealer_count
      my_money = my_money +1;
```

```
      iwin = iwin +1;
   else
      my_money = my_money -1;
      ilose = ilose + 1;
   end%if
   %construct an array that computes a running fraction of losses
   perclose(hand) = ilose/(iwin+ilose+ties);
   end%while
end%for

   iwin
   ilose
   ties
   plot(perclose)
   my_money
-----------------------End of actual file contents--------------------------------

File: dealer_card_value.m
---------------------Start of actual file contents-------------------------

   function newcount = dealer_card_value(card_counter, shuffled_cards, dealer_count)

   card_value = shuffled_cards(card_counter);
   if card_value >= 10 & card_value < 14
        newcount = dealer_count + 10;
   elseif card_value < 10
      newcount = dealer_count + card_value;
   else%card is an ace and has value of 11 for dealer unless he goes bust.
      bigcount = 11 + dealer_count;
      smallcount = 1 + dealer_count;
      if bigcount <22
         newcount = bigcount;
      else
         newcount = smallcount;
      end
   end
-----------------------End of actual file contents--------------------------------

File: dealer_card_value.m
---------------------Start of actual file contents-------------------------

   function newcount = my_card_value(card_counter, shuffled_cards, my_count)

   card_value = shuffled_cards(card_counter);
   if card_value >= 10 & card_value < 14
        newcount = my_count + 10;
   elseif card_value < 10
      newcount = my_count + card_value;
```

```
   else%card is an ace and has value one or 11.
      bigcount = 11 + my_count;
      smallcount = 1 + my_count;
      if bigcount >18 & bigcount <22
         newcount = bigcount;
      else
         newcount = smallcount;
      end
   end
-----------------------End of actual file contents--------------------------------
```