บรรณานุกรม

<u>ภาษาไทย</u>

<u>วิทยานิพนธ์</u>

เจษฎาพร ยุทธนวิบูลย์ชัย. "การศึกษาเปรียบเทียบตัวประมาณริดจ์" วิทยานิพนธ์ ปริญญามหาบัณฑิต ภาควิชาสถิติ บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย, 2533

ทรงพันธุ์ ชุณหสวัสดิกุล. "การประมาณค่าสัมประสิทธิ์การถดถอยพหุโดยที่ค่าประมาณสเกลเปลี่ยนไป" วิทยานิพนธ์ ปริญญามหาบัณฑิต ภาควิชาสถิติ บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย, 2532

ปราณี รัตนัง. "การประมาณสัมประสิทธิ์การถดถอยพหุเมื่อความผิดพลาดมีการแจกแจงแบบเบ้และมีการแจกแจงแบบหางยาวกว่าการแจกแจงปกติ" วิทยานิพนธ์ ปริญญามหาบัณฑิต ภาควิชาสถิติ บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย, 2530

<u>ภาษาต่างประเทศ</u>

<u>หนังสือ</u>

Lee, E.T. <u>Statistical Methods for Survival Data Analysis</u>, Lifetime Learning Publication : USA, 1980.

Robinstein, R.Y. <u>Simulation and Monte Carlo method</u>. New York : John Wiley,1981.

Searle S.R. <u>Linear Models</u>. New York : John Wiley & Sons : 1970

วารสาร

Andrews, D.F. "A Robust Method for Multiple Linear Regression." Technometrics 16 (November 1974) : 523-531.

Gibbons, D.G. "A Simulation Study of Some Ridge Estimators." Journal of the American Statistical Association 76(1981) : 136-139.

Hoerl, A.E. and Kennard, R.W. "Ridge Regression : Biased Estimation for Nonorthogonal Problems." Technometrics 12(1970) : 55-67.

Hoerl, A.E. and Kennard, R.W. and Baldwin, K.F. "Ridge Regression : Some Simulatiion." Communications in Statistics 4(2)(1975) : 105-123.

Lee, T.S. "A Simulation Study of Deterministic Ridge Estimators." Journal of Statistical Computation and Simulation 24(1986) : 171-183

McDonald, G.C. and Galarneau, D.I. "A Monte Carlo Evaluation of Some Ridge tyoe Estimators." Journal of the American Statistical Association 70(1975) : 407-416

Wichern, D.W. and Churchill, A.G. "A Comparison of Ridge Estimators." Technometrics 20(1978) : 301-311

ภาคผนวก

```
{********************************************}
{***     Ridge  Regression  Analysis    ***}
{***                                    ***}
{***   SET  SITUATION IN SIMMULATION    ***}
{********************************************}

PROGRAM  RIDGE_NORM(INPUT,OUTPUT);

CONST
       i_count = 30;          { TOTAL OBSERVATION IN EACH VARIABLE }
       i_vrb = 6;             {    TOTAL INDEPENDENT VARIABLES      }
       i_alpha = 1;           { AMOUNT OF CONST VARIABLE IN MODEL  }

       {************************************************************}
       {***          FOR NORMAL & CONTAMINATE                   ***}
       {***      STD_NORM  =  0.05 , 0.10 AND 0.15              ***}
       {***               FOR LOGNOMAL                          ***}
       {***      STD_NORM  =  0.22 , 0.59 AND 1.00              ***}
       {************************************************************}

       MEAN_NORM  = 1;        { MEAN OF DISTRIBUTION }
       STD_NORM  = 0.05;      { STANDARD DIVATION OF DISTRIBUTION }

       {************************************************************}
       {***               FOR CONTAMINATE NORMAL                ***}
       {***          PERCENT_CONTAMINATE = 5, 10 %              ***}
       {***              SCALE_FACTOR  =  3 AND 10              ***}
       {************************************************************}
       PERCENT_CONTAMINATE = .05;
       SCALE_FACTOR = 3;

       MULTI_CORR = 0.99;     { CORRELATION DURING  X1 TO X3 }
       MULTI_CORR_2 = 0.99;   { CORRELATION BETWEEN X4 WITH X5 }
```

```
     USE_EIGEN = 'MIN';      { BETA FROM EIGENVECTOR FOLLOW EIGENVALUE }
     SIM_TIMES = 200;        { TOTAL TIMES FOR SIMULATION }

     {************************************************************}
     {***                SELECT  DISTRIBUTION                  ***}
     {***   DISTRIBUTION = 1     FOR   NORMAL  DISTRIBUTION    ***}
     {***   DISTRIBUTION = 2     FOR   CONTRAMINATE NORMAL     ***}
     {***   DISTRIBUTION = 3     FOR   LOGNORMAL DISTRIBUTION  ***}
     {************************************************************}
     DISTRIBUTION = 1;
TYPE
     TWO_WAY_1 = ARRAY[1..I_VRB,1..I_VRB]    OF  REAL;
     TWO_WAY_2 = ARRAY[1..I_COUNT,1..I_VRB]  OF  REAL;
     TWO_WAY_3 = ARRAY[1..I_VRB,1..I_COUNT]  OF  REAL;
     ONE_WAY_1 = ARRAY[1..I_VRB]  OF REAL;
     ONE_WAY_2 = ARRAY[1..I_COUNT] OF REAL;

VAR  X_X ,  VX_X , DVX_X,V        :  TWO_WAY_1;
     X                            :  TWO_WAY_2;
     X_T                          :  TWO_WAY_3;
     B   , X_Y  ,X_MEAN, E        :  ONE_WAY_1;
     Y                            :  ONE_WAY_2;
     VALUE_STEP,STACK,SD,K_AVE    :  ARRAY[1..4]  OF REAL;
     I_BELTA ,  I                               :  INTEGER;
     SUM_Y,SUM_YY,V_MSR,V_MSE,SSR               :  REAL;
     IX,IY                                      :  LONGINT;
     MIN_EIGEN , MAX_EIGEN                                  :  REAL;
     POS_MIN_EIGEN,POS_MAX_EIGEN,COUNT_TIMES                :  INTEGER;
     HKB_MSE,TZE_MSE,HK_MSE,BINARY_MSE         : REAL;
     HKB_VAR,TZE_VAR,HK_VAR,BINARY_VAR         : REAL;
     HKB_BIAS,TZE_BIAS,HK_BIAS,BINARY_BIAS     : REAL;
     COUNT_FREQ                                : ARRAY[1..4] OF INTEGER;
```

```
{**************************************************}
{**   ROUTINE   SIMULATE   INDEPENDENT  DATA     **}
{**          DEPENDS  ON  THE  CONDITIONS        **}
{**************************************************}

{*************************************}
{**   SUBROUTINE  FOR  MAKING      **}
{**  R A N D O M   N U M B E R    **}
{*************************************}

PROCEDURE RAND(VAR YFL:REAL);

BEGIN
   REPEAT
      IY:=0;
      IY:=IX*65539;
      IF (IY<0) THEN
          IY := IY+ 2147483647 + 1
      ELSE
          YFL:=IY;
      YFL:=YFL/2147483647;
      IX:=IY;
   UNTIL  (YFL >  0.000001) and (yfl < 1);
END;
```

```
{************************************}
{**   SUBROUTINE  FOR  MAKING      **}
{**     N O R M A L  (0 , 1)       **}
{************************************}

PROCEDURE  NORMAL(VAR  NORM : REAL);

CONST  PI = 3.1415926;

VAR
       RONE,RTWO    : REAL;
       ZONE         : REAL;
       YFL          : REAL;
BEGIN
            ZONE := 0;
            RAND(YFL);
            RONE := YFL;
            RAND(YFL);
            RTWO := YFL;
            ZONE := SQRT(-2*LN(RONE))*COS(2*PI*RTWO);
            NORM := ZONE;
END;
```

```
{*****************************************}
{**   M A I N   S U B R O U T I N E  **}
{**        OF  SIMULATE  DATA          **}
{*****************************************}

PROCEDURE  SIM_NORM;

TYPE   NORM_0_1 = ARRAY[1..I_COUNT] OF REAL;

VAR
       YFL,NORM                  : REAL;
       I,J                       : INTEGER;
       VARIANCE,MEAN,SUM_2       : REAL;
       Z,ZZ                      : NORM_0_1;
       SUMXY, SUMX , SUMY , SUMXX ,  SUMYY , R : REAL;

BEGIN { MAIN PROGRAM }

  FOR I := 1 TO I_COUNT  DO
  BEGIN
      NORMAL(NORM);
      Z[I] := NORM*STD_NORM + MEAN_NORM;
      NORMAL(NORM);
      ZZ[I] := NORM*STD_NORM + MEAN_NORM;
  END;
  FOR I := 2 TO 4  DO
  BEGIN
      MEAN := 0;   SUM_2:= 0;   VARIANCE := 0;
      FOR  J := 1 TO I_COUNT DO
      BEGIN
           NORMAL(NORM);
           X[J,I]  := 0;
           X[J,I]  :=  (1-MULTI_CORR)*NORM + sqrt(MULTI_CORR)*Z[J];
```

```
          MEAN  := MEAN + X[J,I];
          SUM_2 := SUM_2 + SQR(X[J,I]);
      END;
      VARIANCE := SUM_2*I_COUNT - (MEAN*MEAN);
      VARIANCE := VARIANCE/((I_COUNT-1)*I_COUNT);
      VARIANCE := SQRT(VARIANCE);
      MEAN     := MEAN/I_COUNT;
   END;
   SUMXY := 0; SUMX := 0; SUMY := 0; SUMXX := 0;  SUMYY := 0;
   FOR J := 1 TO I_COUNT        DO
   BEGIN
         SUMXY :=  SUMXY + X[J,2] * X[J,3];
         SUMX  :=  SUMX  + X[J,2];
         SUMY  :=  SUMY  + X[J,3];
         SUMXX :=  SUMXX + SQR(X[J,2]);
         SUMYY :=  SUMYY + SQR(X[J,3]);
   END;
   R := 0;
   R :=  I_COUNT * SUMXY  - SUMX*SUMY;
   R :=  R / SQRT((I_COUNT*SUMXX-SQR(SUMX))*(I_COUNT*SUMYY-SQR(SUMY)));
   WRITELN('R = ',R:5:3);

if   I_VRB = 6              THEN
BEGIN
  FOR I := 5 TO 6  DO
  BEGIN
      MEAN := 0;   SUM_2:= 0;   VARIANCE := 0;
      FOR  J := 1 TO I_COUNT DO
      BEGIN
           NORMAL(NORM);
           X[J,I]  := 0;
           X[J,I]  :=  (1-MULTI_CORR_2)*NORM + sqrt(MULTI_CORR_2)*ZZ[J];
           MEAN  := MEAN + X[J,I];
```

```
          SUM_2 := SUM_2 + SQR(X[J,I]);
      END;
      VARIANCE := SUM_2*I_COUNT - (MEAN*MEAN);
      VARIANCE := VARIANCE/((I_COUNT-1)*I_COUNT);
      VARIANCE := SQRT(VARIANCE);
      MEAN     := MEAN/I_COUNT;
   END;
   SUMXY := 0; SUMX := 0; SUMY := 0; SUMXX := 0;  SUMYY := 0;
   FOR J := 1 TO I_COUNT       DO
   BEGIN
         SUMXY :=  SUMXY + X[J,5] * X[J,6];
         SUMX  :=  SUMX  + X[J,5];
         SUMY  :=  SUMY  + X[J,6];
         SUMXX :=  SUMXX + SQR(X[J,5]);
         SUMYY :=  SUMYY + SQR(X[J,6]);
   END;
   R := 0;
   R :=  I_COUNT * SUMXY  - SUMX*SUMY;
   R :=  R / SQRT((I_COUNT*SUMXX-SQR(SUMX))*(I_COUNT*SUMYY-SQR(SUMY)));
   GOTOXY(1,21);WRITELN('R = ',R:5:3);
  END;
 END;
```

```
{****************************************}
{***     Eigen Value & Vectors       ***}
{***                                 ***}
{***       By  Jocobi method         ***}
{****************************************}

PROCEDURE  EIGEN(N:INTEGER);

CONST  EA = 0.0000000001;    {ERROR}
       KM = 100;             {ITERPOLATE}

VAR    A  :  ARRAY[1..6,1..6] OF REAL;
       I,J,KT,I1,J1    :  INTEGER;
       AX,SS,S1,S2,S7,S8,AL,SA              :  REAL;

BEGIN {PROCEDURE}
   for  i  := 1 to n    do
      for  j :=  1 to  n  do
          a[i,j] := 0;   v[i,j] := 0;
      {next j}
   {next i}
   for  i  := 2 to n    DO
   BEGIN
      for j := 2 to n   DO
      BEGIN
          a[i,j] := x_x[i,j];
          v[i,j] := 0;
      END; {NEXT J}
      v[i,i] := 1;
   END; {NEXT I}
   kt := 1;

  REAPEAT
```

```
ax := 0;
for i := 1 to n-1  DO
  for j := i+1 to  n    DO
      if  abs(a[i,j]) > abs(ax)        then
      BEGIN
          i1 := i;
          j1 := j;
          ax := a[i,j];
      end; {END IF}
   {next j}
{next i}
ss := 0;  al :=0;   sa := 0;   s8:= 0;   s7:=0;
if  ax <> 0                      then
BEGIN
   ss := a[i1,i1] - a[j1,j1];
   al := abs(ss)/2;
   sa := sqrt(al*AL + a[i1,j1]*A[I1,J1]);
   s8 := 1 /sqrt(2) * sqrt(1 + al/sa);
   s7 := a[i1,j1] / (2*sa*s8);
   if  ss < 0          then
       s7 := -s7;
   {END IF}
   for i := 1 to n  DO
   BEGIN
     if  (i <> i1)  and  (i <> j1)      then
     BEGIN
         s1 := a[i1,i] * s8 + a[j1,i] * s7;
         s2 := a[j1,i] * s8 - a[i1,i] * s7;
         a[i1,i] := s1;
         a[j1,i] := s2;
     END; {end if}
   END; {next i}
   s1 := a[i1,i1]*s8*S8  +  2*a[i1,j1]*s8*s7  +  a[j1,j1]*s7*S7;
```

```
    s2 := a[i1,i1]*s7*S7  -  2*a[i1,j1]*s8*s7  +  a[j1,j1]*s8*S8;
    a[i1,i1] := s1;
    a[j1,j1] := s2;
    a[i1,j1] := 0;
    a[j1,i1] := 0;
    for  i  := 1 to n     DO
    BEGIN
      a[i,i1] := a[i1,i];
      a[i,j1] := a[j1,i];
      s1 := v[i,i1]*s8 + v[i,j1]*s7;
      s2 := v[i,j1]*s8 - v[i,i1]*s7;
      v[I,i1] := s1;
      v[I,j1] := s2;
   END; {next i}
   kt := kt + 1;
 END; {end if}
 until  (kt >= km) or (abs(ax) <= ea);
MAX_EIGEN := 0.0; MIN_EIGEN := 9E9;
GOTOXY(1,22);
 for i := 2 to n     DO
 BEGIN
    E[i-1] := A[I,I];
     IF  E[I-1] >=  MAX_EIGEN     THEN
     BEGIN
         MAX_EIGEN := E[I-1];
         POS_MAX_EIGEN := I;
     END;
     IF  E[I-1] <=  MIN_EIGEN     THEN
     BEGIN
         MIN_EIGEN := E[I-1];
         POS_MIN_EIGEN := I;
     END;
 WRITE(E[I-1]:12:6);
```

```
      END; {next I}
      IF  USE_EIGEN = 'MIN'           THEN
           FOR I := 2 TO N   DO
               B[I] := V[I,POS_MIN_EIGEN];
      IF  USE_EIGEN = 'MAX'           THEN
           FOR I := 2 TO N   DO
               B[I] := V[I,POS_MAX_EIGEN];

   END; {END SUB}

{**************************************}
{***   FIND VAR & BIAS  ROUTINE    ***}
{**************************************}

PROCEDURE find_var_bias(VAR sk,VARIANCE,BIAS : REAL);

VAR   TRACE,BI : REAL;
      I  : INTEGER;

BEGIN  {PROCEDURE}

    trace := 0 ; bi := 0;
    for i := 1 to i_belta   DO
       trace := trace + e[i]/((e[i]+sk)*(e[i]+sk));
    {next i}
    VARIANCE := v_mse*trace;
    for i := 2 to I_VRB   DO
        bi := bi + b[i]*B[I]/((e[i-1]+sk)*(e[i-1]+sk));
    {next i}
    bias := sk*SK*bi;

END; {end sub}
```

```
{***********************}
{***   HKB  METHOD ***}
{***********************}

PROCEDURE hkb_method(VAR HKB_TOTAL:REAL);

VAR   I,J    : INTEGER;
      HKB,FIND_K_OPT,VARIANCE,BIAS,sk    : REAL;

BEGIN {PROCEDURE}
      find_k_opt := 0;
      for i := 2 to i_vrb    DO
          find_k_opt  := find_k_opt + B[I]*B[I];
      {next i}
      i_belta := i_vrb -1;
      hkB := i_belta*v_mse/find_k_opt;
      sk := hkb;
      K_AVE[1] := K_AVE[1] + HKB;
      FIND_VAR_BIAS(sk,VARIANCE,bias);
      hkb_total := VARIANCE + bias;
      HKB_MSE   := HKB_MSE  + HKB_TOTAL;
      HKB_VAR   := HKB_VAR  + VARIANCE;
      HKB_BIAS  := HKB_BIAS + BIAS;
 END; {END SUB}
```

```
{*******************************}
{***   TZE.SAN LEE  METHOD ***}
{*******************************}

PROCEDURE tze_san_method(VAR TZE_TOTAL:REAL);

VAR  I              : INTEGER;
     V_MIN_EIGEN,VARIANCE,BIAS : REAL;

BEGIN
      v_min_eigen := e[1];
      for i := 2 to i_belta      DO
          if  e[i] < v_min_eigen                then
               v_min_eigen := e[i];
          {end if}
      {next i}
      K_AVE[2] := K_AVE[2] + V_MIN_EIGEN;
      FIND_VAR_BIAS(V_MIN_EIGEN,VARIANCE,bias);
      tze_total := varIANCE + bias;
      TZE_MSE   := TZE_MSE  + TZE_TOTAL;
      TZE_VAR   := TZE_VAR  + VARIANCE;
      TZE_BIAS  := TZE_BIAS + BIAS;
 END; {END SUB}
```

```
{**********************************}
{***   HOERL KENNARD  METHOD ***}
{**********************************}

PROCEDURE Hoerl_Kennard_method(VAR HK_TOTAL:REAL);

VAR   B_MAX,HK,VARIANCE,BIAS  : REAL;
      I       : INTEGER;
BEGIN
      b_max := b[2] * B[2];
      for i := 3 to i_vrb    DO
         if  b_max < b[i]*B[I]        then
             b_max := b[I]*B[I];
        {end if}
     {next i}
      hk := V_mse/b_max;
      K_AVE[3] := K_AVE[3] + HK;
      find_var_bias(hk,VARIANCE,bias);
      hk_total := varIANCE + bias;
      HK_MSE   := HK_MSE  + HK_TOTAL;
      HK_VAR   := HK_VAR  + VARIANCE;
      HK_BIAS  := HK_BIAS + BIAS;
END; {END SUB}
```

```
{********************************}
{***   BINARY SEARCH  METHOD ***}
{********************************}

PROCEDURE binary_method(VAR BINARY_TOTAL:REAL);

CONST  C = 0.0001;

VAR    V_MAX_K,V_MIN_K,OPT_K,SK,VARIANCE,BIAS   :  REAL;
       IJ,I,J : INTEGER;
       STATUS : STRING[10];

BEGIN  {PROCEDURE}

       v_max_k := 1 ; v_min_k := 0 ;
       IJ := 0 ;
       REPEAT
            ij := ij + 1;
            opt_k := (v_max_k + v_min_k) / 2;
            for i := 1 to 3    DO
            BEGIN
               sk := opt_k + (i-2)*c;
               find_var_bias(sk,VARIANCE,bias);
               stack[i] := VARIANCE + bias;
            END; {next I}
            if (Stack[2] <= stack[1]) and (stack[2] <= stack[3])  then
               status := 'stop'
            else if   stack[1] > stack[3]          then
                      v_min_k := opt_k
                else if stack[1] < stack[3]        then
                      v_max_k := opt_k;
            {End if}
```

```
     until ((v_max_k - v_min_k) <= c)  or (status = 'stop');
     K_AVE[4] := K_AVE[4] + OPT_K;
     find_var_bias(opt_k,VARIANCE,bias);
     binary_total := VARIANCE + bias;
     BINARY_MSE   := BINARY_MSE  + BINARY_TOTAL;
     BINARY_VAR   := BINARY_VAR  + VARIANCE;
     BINARY_BIAS  := BINARY_BIAS + BIAS;
END; {END SUB}


{****************************************}
{***   RIDGE  METHOD  ROUTINE     ***}
{****************************************}

PROCEDURE RIDGE_method;

VAR
      HKB_TOTAL,TZE_TOTAL,HK_TOTAL,BINARY_TOTAL  : REAL;

BEGIN {PROCEDURE}


      hkb_method(HKB_TOTAL);
      tze_san_method(TZE_TOTAL);
      Hoerl_Kennard_method(HK_TOTAL);
      binary_method(BINARY_TOTAL);
      IF  HKB_TOTAL <  BINARY_TOTAL        THEN
          COUNT_FREQ[1] := COUNT_FREQ[1] + 1
      ELSE IF TZE_TOTAL  <  BINARY_TOTAL        THEN
              COUNT_FREQ[2] := COUNT_FREQ[2] + 1
           ELSE IF HK_TOTAL <  BINARY_TOTAL        THEN
                   COUNT_FREQ[3] := COUNT_FREQ[3] + 1
                ELSE
```

```
            COUNT_FREQ[4] := COUNT_FREQ[4] + 1;
    {END IF}
    SD[1]   :=  SD[1]   + SQR(HKB_TOTAL);
    SD[2]   :=  SD[2]   + SQR(TZE_TOTAL);
    SD[3]   :=  SD[3]   + SQR(HK_TOTAL);
    SD[4]   :=  SD[4]   + SQR(BINARY_TOTAL);
    GOTOXY(1,5);

    WRITE('HKB = ':15);
    WRITE(HKB_VAR/COUNT_TIMES:10:6,HKB_BIAS/COUNT_TIMES:10:6);
    WRITE(hkb_MSE/COUNT_TIMES:10:6);
    WRITE((COUNT_TIMES*SD[1] - SQR(HKB_MSE))/COUNT_TIMES:12:6);
    WRITELN(COUNT_FREQ[1]:9,K_AVE[1]/COUNT_TIMES:13:4  );

    WRITE('Tze-San Lee = ':15);
    WRITE(TZE_VAR/COUNT_TIMES:10:6,TZE_BIAS/COUNT_TIMES:10:6);
    WRITE(tze_MSE/COUNT_TIMES:10:6);
    WRITE((COUNT_TIMES*SD[2] - SQR(TZE_MSE))/COUNT_TIMES:12:6);
    WRITELN(COUNT_FREQ[2]:9,K_AVE[2]/COUNT_TIMES:13:4  );

    WRITE('HK = ':15);
    WRITE(HK_VAR/COUNT_TIMES:10:6,HK_BIAS/COUNT_TIMES:10:6);
    WRITE(hk_MSE/COUNT_TIMES:10:6);
    WRITE((COUNT_TIMES*SD[3] - SQR(HK_MSE))/COUNT_TIMES:12:6);
    WRITELN(COUNT_FREQ[3]:9,K_AVE[3]/COUNT_TIMES:13:4  );

    WRITE('BINARY = ':15);
    WRITE(BINARY_VAR/COUNT_TIMES:10:6,BINARY_BIAS/COUNT_TIMES:10:6);
    WRITE(BINARY_MSE/COUNT_TIMES:10:6);
    WRITE((COUNT_TIMES*SD[4] - SQR(BINARY_MSE))/COUNT_TIMES:12:6);
    WRITELN(COUNT_FREQ[4]:9,K_AVE[4]/COUNT_TIMES:13:4 );
END; {END SUB}
```

```
{************************************************}
{***  SUBROUTINE READ DATA & SET PARAMETER  ***}
{************************************************}

PROCEDURE VALUE_IN_MATRIC_RTN;

var
     i,j,I_PP,R1,R2,C1,C2,K : integer;
     i_belta:integer;
     SYY  : REAL;
     norm :real;

begin {PROCEDURE}

     i_belta:=i_vrb-i_alpha;
     FOR J := 1 TO  I_VRB   DO
        X_MEAN[J] := 0;
     {NEXT J}
     for i:=1 to i_count do
      begin
        for j:=1 to i_vrb do
         begin
           if j=1 then
            begin
              x[i,j] :=1;
              x_t[j,i]:=1;
            end
           else
            begin
              x_t[j,i]:=x[i,j];
            end; {END IF}
           x_mean[j]:=x[i,j]/i_count+x_mean[j];
          end; {NEXT J}
```

```
end;{NEXT I}

{*********  Minus Matrix X  with Mean  ****************}

        for i := 1 to i_count      do
        begin
           for j := 2 to i_vrb       do
           begin
                x[i,j] := x[i,j] - x_mean[j];
                x_t[j,i] := x[i,j];
           end;
        end;

{************  Build Matrix X'X  ******************}

      r1 := i_vrb ; c1 := i_count;
      c2 := i_vrb ; r2 := i_count;
      for i  := 1 to r1  DO
        for j  :=1 to c2    DO
        begin
            x_x[i,j] := 0;
          for k := 1 to c1    DO
              x_x[i,j] := x_t[i,k] * x[k,j] + x_x[i,j];
          {next k}
        end; {next j}
       {next i}

       EIGEN(I_VRB);
```

```
{*******************************}
{**  M A T R I X    Y       **}
{*******************************}

        sum_y  := 0;  sum_yy := 0; syy:= 0;
        for j := 1 to I_count   DO
        BEGIN
            y[j]  := 0;
            for i := 2 to i_vrb   do
            begin
               y[j]  :=  y[j] +  b[i]*x[j,i];
            end;
             normal(norm);
             norm := norm*std_norm + mean_norm;
             y[J] := y[J] + norm;
            sum_y  := sum_y  + y[j];
            sum_yy := sum_yy + y[j]*Y[J];
       end;

        syy := sum_yy - (sum_y)*(SUM_Y)*I_alpha / I_count;


{***********  Build Matrix X'Y  ******************}

          r1 := i_vrb ; c1 := i_count;
          c2 := 1      ; r2 := i_count;
       for i  := 1 to r1  DO
       begin
         x_y[i] := 0;
           for k := 1 to c1    DO
           begin
               x_y[i] := x_t[i,k] * y[k] + x_y[i];
           end;
```

```
          {next k}
         {next j}
       end; {next i}

END; {end sub}

{****************************************}
{***  SUBROUTINE  INVERSE  MATRIX    ***}
{****************************************}

PROCEDURE  INVERSE_MATRIC_RTN;

VAR    I,J,K,L,M,N,O,Q : INTEGER;
       RATIO,TEMP : REAL;
       DX_X  : ARRAY[1..5,1..10] OF REAL;

BEGIN {PROCEDURE}

   N := 2*I_VRB;
   FOR I := 1 TO 5  DO
      FOR J:= 1 TO 10 DO
        DX_X[I,J] :=0.0;
      {NEXT J}
    {NEXT I}
   for i := 1 to I_vrb   DO
   BEGIN
      for j := 1  to I_vrb   DO
        dX_x[i,j]  :=  x_x[i,j];
      {NEXT J}
      DX_X[I,J+I] := 1.0;
   END;

   FOR I := 1 TO I_VRB DO
```

```
BEGIN

    K:=0;
    IF  DX_X[I,I] <> 0.0     THEN
       BEGIN
         IF DX_X[I,I] <> 1.0   THEN
         BEGIN
             RATIO := 1/DX_X[I,I];
             FOR J := I TO N  DO
                DX_X[I,J] := DX_X[I,J] * RATIO
         END;
         FOR J := 1 to i_vrb  DO
            IF (J <> I) AND (DX_X[J,I] <> 0.0)    THEN
            BEGIN
               RATIO := DX_X[J,I];
               FOR L := I TO N   DO
               BEGIN
                   DX_X[J,L] := DX_X[J,L] - DX_X[I,L]*RATIO;
                   IF  ABS(DX_X[J,L]) < 1E-07      THEN
                       DX_X[J,L] := 0.0;
                END
            END
       END
    ELSE
       BEGIN
         K := K +1;
         J := 0;
         WHILE J < I_VRB  DO
         BEGIN
            J := J + 1;
            IF (DX_X[J,I] <>0.0) OR (J>I_VRB)  THEN
               J:=I_VRB
            ELSE
```

```
        K:=K+1;
    END;
    J:=0;
    IF  K <= I_VRB-I        THEN
    BEGIN
        FOR J := I TO N    DO
        BEGIN
           TEMP := DX_X[I,J];
           DX_X[I,J] := DX_X[I+K,J];
           DX_X[I+K,J]  := TEMP;
        END;
        IF DX_X[I,I] <> 1.0   THEN
        BEGIN
            RATIO := 1/DX_X[I,I];
             FOR J := I TO N  DO
                 DX_X[I,J] := DX_X[I,J] * RATIO;
        END;
        FOR J := 1 to i_vrb  DO
          IF (J <> I) AND (DX_X[J,I] <> 0.0)    THEN
          BEGIN
               RATIO := DX_X[J,I];
               FOR L := I TO N   DO
               BEGIN
                  DX_X[J,L] := DX_X[J,L] - DX_X[I,L]*RATIO;
                  IF  ABS(DX_X[J,L]) < 1E-07       THEN
                      DX_X[J,L] := 0.0;
               END
          END
     END
   END;
{END IF}

WRITE('LOOP --> ',I,' ':5);
```

```
      FOR Q := 1 TO I_VRB  DO
      BEGIN
         FOR J := 1  TO N  DO
             WRITE(DX_X[Q,J]:10:3);
         WRITELN;WRITE('':15);
      END;
      WRITELN;
  END;
  {NEXT I}
         FOR I := 1 TO I_VRB   DO
         BEGIN
           FOR J := I_VRB+1  TO N  DO
               WRITE(DX_X[I,J]:10:3);
             WRITELN;
         END;
END;


{****************************************}
{***  SUBROUTINE  FIND EQUATION    ***}
{****************************************}

PROCEDURE CROSS_MATRIC_RTN;

VAR I,J,K : INTEGER;
    B_CONST : REAL;

BEGIN
        b_const := 0; v_mar := 0; v_mae := 0;
       for  i := 1 to  I_vrb   DO
       BEGIN
          b[I] := 0;
```

```
    for  j  := 1  to  I_vrb    DO
      b[i] :=  b[i] + Vx_x[i,j] * x_y[j];
    {next  j}
  END;  {NEXT I}
  b_const := sum_y/i_count;
  for i := 2 to i_vrb   DO
      b_Const := b_const - b[i]*x_mean[i];
  {next i}
  ssr :=0;
  FOR I := 1 TO I_vrb   DO
    SSR :=  SSR + B[I]* X_Y[I];
  {NEXT I}
  v_MSR := SSR/I_vrb;
  v_MSE := (Sum_yy-SSR)/(I_COUNT-I_vrb);

END; {END SUB}
```

```
{*******************************************}
{***  SUBROUTINE PRINT ANOVA FOR OLS   ***}
{*******************************************}

PROCEDURE PRINT_RESULT_RTN;

BEGIN {PROCEDURE}

 WRITELN('');
 WRITELN('ANOVA':30);
 WRITELN('---------------------------------------------------------------');
 WRITELN('   SOV      DF          SS                    MS              F');
 WRITELN('---------------------------------------------------------------');
   WRITE('  SSR');
   WRITE(I_vrb:9);
   WRITE(SSR:20:3);
   WRITE(V_MSR:15:3);
 WRITELN(V_MSR/V_MSE:13:3);
   WRITE('  SSE');
   WRITE(I_COUNT-I_vrb:9);
   WRITE(SUM_YY-SSR:20:3);
 WRITELN(V_MSE:15:3);
 WRITELN('---------------------------------------------------------------');
   WRITE('  TOTAL');
   WRITE(I_COUNT:6);
 WRITELN(SUM_YY:20:3);
 WRITELN('---------------------------------------------------------------');

END; {END SUB}
```

```
{*************************}
{***  MAIN  ROUTINE  ***}
{*************************}


BEGIN
  clrscr;
  HKB_MSE := 0; TZE_MSE := 0; HK_MSE := 0; BINARY_MSE := 0;
  HKB_VAR := 0; TZE_VAR := 0; HK_VAR := 0; BINARY_VAR := 0;
  HKB_BIAS:= 0; TZE_BIAS:= 0; HK_BIAS:= 0; BINARY_BIAS:= 0;
  FOR  I  := 1 TO 4    DO
  BEGIN
       COUNT_FREQ[I] := 0;
       SD[I]         := 0;
       K_AVE[I]      := 0;
  END;
  WRITELN('METHOD':12,'AVAR':12,'ABIAS':10,'AMSE':10,'SD':10,
          'TIMES(MIN)':17,'K':5);
  IX := 116;
  FOR  COUNT_TIMES  :=  1  TO  SIM_TIMES  DO
  BEGIN
       writeln('Round = ',count_times);
       SIM_NORM;
       VALUE_IN_MATRIC_RTN;
       INVERSE_MATRIC_RTN;
       CROSS_MATRIC_RTN;
       Ridge_method;
  END; {NEXT COUNT_TIMES}
      WRITELN('');
      WRITE('RDTMSE between hkb  and Binary = ':40);
      WRITELN((hkb_MSE - binary_MSE)/binary_MSE * 100:6:3 ,' %');
      WRITE('RDTMSE between Tze-San Lee and Binary = ':40);
```

```
WRITELN((tze_MSE - binary_MSE)/binary_MSE * 100:6:3 ,' %');
WRITE('RDTMSE between hk  and Binary = ':40);
WRITELN((hk_MSE - binary_MSE)/binary_MSE * 100:6:3  ,' %');

WRITELN;WRITELN;
WRITELN('NORMAL ( MEAN = ',MEAN_NORM,' STD = ',STD_NORM:5:3,' )');
WRITELN('AMOUNT OF INDEPENDENT VARIABLE =  ',I_VRB-1);
WRITELN('AMOUNT OF OBSERVATIONS          =  ',I_COUNT);
WRITELN('AMOUNT OF MULTICOLINEARITY      =  ',MULTI_CORR:5:3);
WRITELN('BELTA FROM EIGEN                =  ',USE_EIGEN);

END.
```

ประวัติผู้เขียน

นายจิรายุส พู่มนตรี เกิดเมื่อวันที่ 1 มิถุนายน พ.ศ. 2509 จบการศึกษาระดับปริญญาตรีจากสถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหาร ลาดกระบัง สาขาสถิติประยุกต์ (เกียรตินิยมอันดับ 2) คณะวิทยาศาสตร์ เมื่อปี พ.ศ. 2530 ได้เข้าศึกษาในภาควิชาสถิติ คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย เมื่อปี พ.ศ. 2531 ปัจจุบันเป็นอาจารย์ประจำศูนย์วิจัยธุรกิจ มหาวิทยาลัยอัสสัมชัญ