รายการอ้างอิง

ภาษาไทย

กัลยา วานิชย์บัญชา. การใช้ SPSS for Windows ในการวิเคราะห์ข้อมูล เวอร์ชั่น 7 – 10. พิมพ์ครั้งที่ 2. กรุงเทพมหานคร : ซี เค แอนด์ เอส โฟโต้สตูดิโอ. 2543.

จินดา ยาปนเวช. คู่มือเรียนภาษา Pascal. พิมพ์ครั้งที่ 1. กรุงเทพมหานคร : โปรวิชั่น. 2545.

จิตรวี วีระประดิษฐ์. การประมาณค่าพารามิเตอร์ในสมการถดถอยเชิงเส้นพหุ เมื่อข้อมูลมีค่าผิดปกติ. วิทยานิพนธ์ปริญญามหาบัณฑิต สาขาวิชาสถิติ ภาควิชาสถิติ คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย. 2538.

ประชุม สุวัตถี. ทฤษฎีการอนุมานเชิงสถิติ. พิมพ์ครั้งที่ 2. กรุงเทพมหานคร : โรงพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย. 2545.

มนตรี พิริยะกุล. ทฤษฎีสถิติ 2 : Theory of Statistics 2 (ST412). พิมพ์ครั้งที่ 6. กรุงเทพมหานคร : สำนักพิมพ์มหาวิทยาลัยรามคำแหง. 2536.

มนตรี พิริยะกุล. เทคนิคการวิเคราะห์สมการถดถอย : Regression Analysis (ST331). พิมพ์ครั้งที่ 1. กรุงเทพมหานคร : สำนักพิมพ์มหาวิทยาลัยรามคำแหง. 2536.

มานพ วราภักดิ์. การจำลองเบื้องต้น : Introduction to Simulation. กรุงเทพมหานคร : ศูนย์ผลิตตำราเรียน สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ. 2547.

สุชาดา กีระนันทน์. การอนุมานเชิงสถิติ : ทฤษฎีขั้นต้น. พิมพ์ครั้งที่ 3. กรุงเทพมหานคร : โรงพิมพ์จุฬาลงกรณ์มหาวิทยาลัย. 2534.

สุพล ดุรงค์วัฒนา. การวิเคราะห์เชิงสถิติ : การวิเคราะห์ความถดถอย. กรุงเทพมหานคร.โรงพิมพ์จุฬาลงกรณ์มหาวิทยาลัย. 2536.

ภาษาต่างประเทศ

Askin, R. G., and Montgomery, D. C. Augmented robust estimaters. Technometrics 22 : 333 - 341. 1980.

Casella, G. and Berger, R. L. Statistical Inference. 2nd ed. Pacific Grove : Duxbury. 2002.

Chan,L.K. & Mak,T.K. On the polynomial functional relationship. Journal of the Royal Statistical Society Series B,47,510-518. 1985.

C.Cheng & H.Schneeweiss. The polynomial regression with errors in the variables.Journal of the Roval Statistical Society Series B.60,189-199. 1998.

Danodar N.Gujarati. Basic Econometrics. : McGraw Hill. 1995.

G.S.Maddala. Econometrics. Florida : McGraw Hill. 1997

Hamilton, L. C. Modern data analysis : A first course in applied statistics. California : Wadsworth : 116 – 147. 1990.

Hoerl, A. E., and Kennard, R. W. Ridge Regression : Applications to nonorthogonal problems. Technometrics 12 : 69 – 82. 1970.

Hoerl, A. E., and Kennard, R. W., and Bladwin, K. F. Ridge Regression : Some simulations. Commun. Stat. 4 : 105 – 123. 1975.

Pfaffenberger, R. C., and Dielman, T. E. Robust regression : analysis and applications. New York : Marcel Dekker : 243 – 270. 1990.

Pfaffenberger, R. C., and Dielman, T. E. A comparison of robust ridge estimators. Proceedings of the American Statistical Association Business and Economic Statistics Section. Las Vegas , Nev., : 631 – 635. 1985.

Pfaffenberger, R. C., and Dielman, T. E. A modified ridge regression estimator using the least absolute value criterion in the multiple linear regression model. Proceedings of the American Institue for Decision Sciences. Toronto, : 791 – 793. 1984.

S.Huang & L Huwang. On the polynomial structural relationship. The Canadian Journal Statistics.29.465-512. 2001.

ภาคผนวก

ตารางแสดงลักษณะการทำงานของโปรแกรมทั้งหมดที่ใช้การวิจัย

อันดับที่	ชื่อโปรแกรม	การทำงานของโปรแกรม	ชื่อโปรแกรมย่อยที่เรียกใช้
โปรแกรมหลัก	Source_thesis	- สร้างข้อมูลของตัวแปรอิสระ ค่าคลาดเคลื่อนสุ่ม ค่าคลาดเคลื่อนในตัวแปรอิสระและตัวแปรตาม	get_x , get_y , poly_x , get_error
		- คำนวณค่า *RRMSE* ของวิธี OLS	get_mse_beta
		- คำนวณค่า *RRMSE* ของวิธี ROLS	
		- คำนวณค่า *RRMSE* ของวิธี RLAV	
โปรแกรมย่อย			
1	datanormal	- การสร้างเลขสุ่มให้มีการแจกแจงแบบปกติ	
2	OLS	- การประมาณค่าพารามิเตอร์ที่เหมาะสมโดยวิธี OLS	xtxcross, xtycross , inv , estimat_beta, get_mse_beta
3	ROLS	- การประมาณค่าพารามิเตอร์ที่เหมาะสมโดยวิธี ROLS	get_var ,get_k , xtxplusk , inv , estimate_beta, get_mse_beta
4	RLAV	- การประมาณค่าพารามิเตอร์ที่เหมาะสมโดยวิธี RLAV	leas_abs , get_var , get_k , xtxplusk ,inv, estimate_beta , get_mse_beta
5	get_x	- สร้างตัวแปรอิสระ	
6	poly_x	- สร้างตัวแปรอิสระที่มีการยกกำลังตามที่ต้องการ	datanormal

อันดับที่	ชื่อโปรแกรม	การทำงานของโปรแกรม	ชื่อโปรแกรมย่อยที่เรียกใช้
7	get_error	- สร้างความคลาดเคลื่อนสุ่ม	datanormal
8	get_y	- สร้างตัวแปรตาม	
9	get_beta	- สร้างค่าสัมประสิทธิ์การถดถอย	
10	xtxcross	- คำนวณ x ทรานโพส x	
11	xtycross	- คำนวณ x ทรานโพส y	
12	inv	- คำนวณเมทริกซ์ผกผัน	det
13	estimate_beta	- คำนวณค่าสัมประสิทธิ์การถดถอย	
14	leas_abs	- คำนวณค่าสัมประสิทธิ์การถดถอยวิธี least absolute value	
15	btbcross	- คำนวณค่า beta ทรานโพส beta	
16	xtxplusk	- คำนวณเมทริกซ์ที่บวกค่า k ในแนวทะแยงของเมทริกซ์	
ฟังก์ชัน			
19	det	- คำนวณค่าดีเทอร์มิแนน	
20	get_var	- คำนวณค่าความแปรปรวน	
21	get_k	- คำนวณค่า k ของริดจ์	
22	get_mse_beta	- คำนวณค่า MSE	

โดยรายละเอียดของโปรแกรมมีดังนี้

```
program source_thesis;
{$APPTYPE CONSOLE}
uses
 Math,  SysUtils;
const
        sample1 = 15; sample2 = 30; sample3 = 50; sample4 = 100; sample5 = 200;
        power1 = 2; power2 = 3; power3 = 4; power4 = 5; power5 = 6; iteration = 1,000;
        varian1 = 4; varian2 = 6; varian3 = 8; varian4 = 10;
type
        data = array[1..sample5,1..(power5+1)]of double;
        data2 = array[1..sample5]of double;
        data3 = array[1..(power5+1)]of double;
        data4 = array[1..(power5+1),1..(power5+1)]of double;
        data5 = array[1..iteration]of double;
        data6 = array[1..((2*(power5+1))+(3*sample5))]of double;
        data7 = array[1..(sample5+1),1..((2*(power5+1))+(3*sample5)+1)]of double;
        data8 = array[1..(2*(power5+1))]of integer;
        data9 = array[1..iteration]of double;
var
        a,aa,b,c,s,t1, varian,n,p,t,count:integer;
        amse_ols,amse_rid_ols,amse_rid_lav, cv_ols,cv_rid_ols,cv_rid_lav:double;
        sigma_ols,sigma_rid_ols,sigma_rid_lav, rmse_beta_ols,rmse_beta_rid_ols,rmse_beta_rid_lav:double;
        mse_ols,mse_rid_ols,mse_rid_lav:data9;
        amse_beta_ols,amse_beta_rid_ols,amse_beta_rid_lav, s_ols,k_ols,s_lav,k_rid_lav,test,amse_min:double;
        x_model,x_ols:data;
        e,z,y_ori,x_ori,xb_ols,xb_rid_ols,xb_rid_lav:data2;
        beta,beta_ols,beta_rid_ols,beta_lav,beta_rid_lav:data3;
        xty_ols,xty_rid_ols,xty_rid_lav:data3;
        x_inv_ols,x_inv_rid_ols,x_inv_rid_lav,xtx_ols,xtx_rid_ols,xtx_rid_lav:data4;
procedure datanormal(n:integer;var z:data2);
var i:integer; r1,r2:double; z1:data2;
begin
  randomize;
  for i:= 1 to sample5 do
    z1[i]:= 0;
  i := 1;
  repeat
    if (i <= n)then
      begin
        r1 := random;  r2 := random;
        z1[i]:=sqrt((-2)*ln(r1))*(cos(2*pi*r2));
```

```
        end
      else
          z1[i]:= 0;
          i := i+1;
        until i > sample5;
  z := z1;
end;
procedure get_error(n:integer; sigmae:double; var e:data2);
var i:integer; z2,temp:data2;
begin
  for i:= 1 to sample5 do
  begin
    z2[i]:= 0;
    temp[i]:= 0;
  end;
  datanormal(n,z2);
  for i:= 1 to sample5 do
  begin
    if (i <= n) then
      temp[i]:= sqrt(sigmae)*z2[i]
    else
      temp[i]:= 0;
  end;
  e:=temp;
end;
procedure get_x(n:integer; var x:data2);
var i1,k1:integer; mu,sigma:double; x1:data; nor,z3:data2;
begin
  for i1:= 1 to sample5 do
  begin
    z3[i1]:= 0;
    nor[i1]:= 0;
  end;
  datanormal(n,z3);
  mu := 2;
  sigma := 1;
  for k1:= 1 to n do
    nor[k1]:= mu+(sigma*z3[k1]);
  x:=nor;
end;
procedure poly_x(n,p:integer;x_in:data2;var x_out:data);
```

```
var i1,j1:integer; x1:data;
begin
  for i1:= 1 to sample5 do
    for j1:= 1 to (power5+1) do
      x1[i1,j1]:=0;
  for i1:= 1 to sample5 do
  begin
    for j1:= 1 to p+1 do
    begin
      if (i1<=n) then
      begin
        if (j1=1) then
          x1[i1,j1]:= 1
        else
          x1[i1,j1]:= power(x_in[i1],(j1-1));
      end
      else
        x1[i1,j1]:=0;
    end;
  end;
  x_out:=x1;
end;
procedure xtxcross(x:data;n,p:integer;var xtx:data4);
var i,j,k:integer;
begin
  for i:=1 to p do
    for j:= 1 to p do
    begin
      xtx[i,j] := 0;
      for k := 1 to sample5 do
        xtx[i,j] := xtx[i,j] + (x[k,i]*x[k,j])
    end;
end;
procedure get_beta(p:integer; var beta:data3);
var i:integer;
begin
  for i:= 1 to (power5+1) do
  begin
    if (i<=(p)) then
      beta[i] := 1
    else
```

```
      beta[i] := 0
  end;
end;
procedure get_y(x:data;b:data3;e:data2;n:integer; var y:data2);
var i,j : integer;
begin
  for i:=1 to sample5 do
  begin
    y[i]:= 0;
    for j:=1 to (power5+1) do
      y[i]:= y[i]+ (x[i,j]*b[j]);
    y[i]:= y[i]+ e[i];
  end;
end;
function det(x:data4;p:integer):double;
var i,j,k:integer; temp,temp1:data4; sum:double;
begin
  if (p = 2) then
    det:= (x[1,1]*x[2,2])-(x[2,1]*x[1,2])
  else
  begin
    sum:= 0;
    for j:= 1 to p-1 do
      for k:= 1 to p do
          temp[j,k]:= x[j+1,k];
    temp1:=temp;
    for i:= 1 to p do
    begin
      if i=1 then
      begin
        for j:= 1 to p-1 do
          for k:= 1 to p-1 do
            temp[k,j]:= temp1[k,j+1]
      end
      else
      begin
        for j:= 1 to p-1 do
          for k:= 1 to p-1 do
            if j < i then
              temp[k,j]:= temp1[k,j]
            else
```

```
            temp[k,j]:= temp1[k,j+1];
        end;
        sum:=sum+(power((-1),(i+1))*x[1,i]*det(temp,p-1));
      end;
      det := sum;
    end;
end;
procedure inv(x:data4; p:integer; var xin:data4);
var i,j,kr,kc,m:integer; d,cal:double; temp:data4;
begin
  for i:=1 to p do
    for j:=1 to p do
      xin[i,j]:=0;
  d:= det(x,p);
  if(d<>0)then
  begin
    for i:= 1 to p do
    begin
      for j:=1 to p do
      begin
        if (i=1) then
        begin
          for kr:= 1 to p-1 do
            for kc:=1 to p do
              temp[kr,kc]:=x[kr+1,kc];
          if (j=1) then
            for kc:= 1 to p-1 do
              for kr:= 1 to p-1 do
                temp[kr,kc]:=temp[kr,kc+1]
          else
            for kc:= 1 to p-1 do
              for kr:= 1 to p-1 do
                if (kc<j) then
                  temp[kr,kc]:=temp[kr,kc]
                else
                  temp[kr,kc]:=temp[kr,kc+1];
        end
        else
        begin
          for kr:= 1 to p-1 do
            for kc:= 1 to p do
```

```
            if (kr<i) then
              temp[kr,kc]:=x[kr,kc]
            else
              temp[kr,kc]:=x[kr+1,kc];
          if (j=1) then
            for kc:= 1 to p-1 do
              for kr:= 1 to p-1 do
                temp[kr,kc]:=temp[kr,kc+1]
          else
            for kc:= 1 to p-1 do
              for kr:= 1 to p-1 do
                if (kc<j) then
                  temp[kr,kc]:=temp[kr,kc]
                else
                  temp[kr,kc]:=temp[kr,kc+1];
        end;
          xin[j,i]:= power((-1),(i+j))*(1/d)*(det(temp,p-1));
      end;
    end;
  end
  else
    writeln('Undefined Invert of XTX');
end;
procedure xtycross(x:data; y:data2; n,p:integer; var xy:data3);
var i,j:integer;
begin
  for i:= 1 to power5+1 do
    xy[i]:=0;
  for i:=1 to p+1 do
  begin
    for j:=1 to n do
      xy[i]:= xy[i]+(x[j,i]*y[j]);
  end;
end;
procedure estimat_beta(x:data4;y:data3;p:integer;var beta:data3);
var i,j:integer;
begin
  for i:=1 to (p+1) do
  begin
    beta[i]:=0;
    for j:= 1 to (p+1) do
```

```
        beta[i]:= beta[i]+(x[i,j]*y[j]);
    end;
  end;
procedure xbcross(x:data;b:data3;n,p:integer;var xb:data2);
var i,j:integer;
begin
  for i:= 1 to n do
  begin
    xb[i]:=0;
    for j:= 1 to p do
      xb[i]:=xb[i]+(x[i,j]*b[j]);
  end;
end;
function get_mse_beta(beta:data3;degree:integer):double;
var i:integer; temp1:double; set_beta:data3;
begin
  for i:= 1 to degree do
    set_beta[i]:= 1;
  temp1:=0;
  for i:= 1 to degree do
    temp1:= temp1 + power((set_beta[i]-beta[i]),2);
  get_mse_beta:=temp1/degree;
end;
procedure leas_abs(x:data; y:data2; nn,pp:integer; var b:data3);
var m,m2 :integer; flag:data8;
   n,n2,nn1,nn2,nn3,nn4,nn5,nn7:integer;
   i,j,k,l,kr,p,times,e1,f:integer; cc:data6; cr,ik,e:data2;
   a:data7; am,aa,ec:double;
   canin,canout:boolean; bp,bm:data3;
begin
  m:= nn;
  p:=2*(pp+1);
  n:=(3*nn)+p;
  m2 := m+1;
  n2:=n+1;
  for i:= 1 to p do cc[i]:=0;
  nn3:= (2*nn)+p;
  for i:= p+1 to nn3 do cc[i]:=1;
  nn4:= nn3+1;
  nn5:=(3*nn)+p;
  for i:= nn4 to nn5 do cc[i]:= 1000;
```

```
for i:= 1 to nn do cr[i]:= 1000;
for i:= 1 to nn do ik[i]:= nn4+i-1;
for i:= 1 to nn do
begin
  j:= 1;
  repeat
    k:= ((2*j)-1);
    a[i,k]:=x[i,j];
    k:=k+1;
    a[i,k]:= -x[i,j];
    j:=j+1;
  until (j>pp+1);
  nn1:=nn+p;
  for j:= (p+1) to nn1 do
    if((j-i)=p)then
      a[i,j]:=1
    else
      a[i,j]:=0;
  nn2:=nn1+1;
  for j:= nn2 to nn3 do
    if((j-i)=nn1)then
      a[i,j]:= -1
    else
      a[i,j]:=0;
  for j:= nn4 to nn5 do
    if((j-i)=nn3)then
      a[i,j]:=1
    else
      a[i,j]:=0;
  nn7:=nn5+1;
  a[i,nn7]:=y[i];
  if y[i]<0 then
    for j:= 1 to nn7 do a[i,j]:= -a[i,j];
end;
for j:= 1 to n do
begin
  a[m2,j]:=0;
  for l:= 1 to m do
    a[m2,j]:=(cr[l]*a[l,j]);
end;
for j:= 1 to n do
```

```
  a[m2,j]:=cc[j]-a[m2,j];
a[m2,n2]:=0;
for i:= 1 to m do
  a[m2,n2]:=a[m2,n2]+(cr[i]*a[i,n2]);
repeat
  canin:=false;
  for j:= 1 to n do
    if(a[m2,j]<0)then
    begin
      canin:=true;
      break;
    end;
  if canin then
  begin
    k:=1;
    am:=a[m2,1];
    for j:=2 to n do
      if(a[m2,j]<am)then
      begin
        am:=a[m2,j];
        k:=j;
      end;
    canout:=false;
    for i:= 1 to m do
      if(a[i,k]>0.000001)then
      begin
        canout:=true;
        break;
      end;
    if canout then
    begin
      for i:= 1 to m do
        if(a[i,k]>0)then
          e[i]:=a[i,n2]/a[i,k]
        else
          e[i]:=99999.9;
      for i:= 1 to m do
        if (e[i]>=0)then
        begin
          kr:=i;
          ec:=e[i];
```

```
            break;
          end;
        for i:= 1 to m do
          if((0<=e[i])and(e[i]<ec))then
          begin
            ec:=e[i];
            kr:=i;
          end;
        ik[kr]:=k;
        cr[kr]:=cc[k];
        for i:= 1 to m2 do
        begin
          if(i<>kr)then
          begin
            for j:= 1 to n2 do
              if(j<>k)then
                a[i,j]:=a[i,j]-((a[i,k]*a[kr,j])/a[kr,k]);
            a[i,k]:=0;
          end;
        end;
        aa:=a[kr,k];
        for j:= 1 to n2 do
          a[kr,j]:=a[kr,j]/aa;
        a[m2,n2]:=0;
        for i:= 1 to m do
          a[m2,n2]:= a[m2,n2]+(cr[i]*a[i,n2]);
      end;
    end;
  until((not canin)or(not canout));
  for i:= 1 to p do
    flag[i]:=0;
  for i:= 1 to m do
    for j:= 1 to p do
      if(ik[i]=j)then
        flag[j]:=i;
  i:=1;
  j:=0;
  repeat
    if(flag[i+j]<>0)then
      bp[i]:=a[flag[i+j],n2]
    else
```

```
      bp[i]:=0;
    if(flag[i+j+1]<>0)then
      bm[i]:=a[flag[i+j+1],n2]
    else
      bm[i]:=0;
    i:= i+j;
    j:= j+1;
  until (i>(pp+1));
  for i:= 1 to pp+1 do
    b[i]:=bp[i]-bm[i];
end;
function get_var(x:data;y:data2;b:data3;n,p:integer):double;
var i:integer; tmp:double; xb:data2;
begin
  xbcross(x,b,n,p,xb);
  tmp:=0;
  for i:= 1 to n do
    tmp := tmp + ((y[i]-xb[i])*(y[i]-xb[i]));
  tmp:=tmp/(n-p);
  get_var:=tmp;
end;
procedure btb_cross(bt:data3;n:integer;var btb:double);
var i:integer;
begin
  btb := 0;
  for i:= 1 to n do
    btb:=btb+(bt[i]*bt[i]);
end;
function get_k(p:integer;s:double;b:data3):double;
var btb,temp:double;
begin
  temp:=0;
  btb_cross(b,p,btb);
  temp:=(p*s)/btb;
  get_k:=temp;
end;
procedure xtxplusk(xtx:data4;p:integer;k:double;var xtxk:data4);
var i,j:integer;
begin
  for i:= 1 to p do
    for j:= 1 to p do
```

```
        if(i=j)then
          xtxk[i,j]:=xtx[i,j]+k
        else
          xtxk[i,j]:=xtx[i,j];
end;
begin {Main Program}
 { TODO -oUser -cConsole Main : Insert code here }
  writeln;
  randomize;
  for a:= 1 to 5 do
  begin
    case a of
      1 : n:= sample1;   2 : n:= sample2;   3 : n:= sample3;   4 : n:= sample4;    else n:= sample5;
    end;
    get_x(n,x_ori);
    for b:=1 to 5 do
    begin
      case b of
        1 : p:= power1;   2 : p:= power2;   3 : p:= power3;   4 : p:= power4;   else p:= power5;
      end;
      for s:= 1 to n do
        for t1:= 1 to p+1 do
          x_ols[s,t1]:= 0;
      poly_x(n,p,x_ori,x_model);
      for c:= 1 to 4 do
      begin
        case c of
          1 : varian:=varian1;   2 : varian:=varian2;   3 : varian:=varian3;    else varian:=varian4;
        end;
        writeln(' sample size = ',n:3,' degree = ',p:2,' variance model = ',varian:3);
        writeln;
        amse_ols:= 0;   amse_rid_ols:= 0;   amse_rid_lav:= 0;    rmse_beta_ols:= 0; rmse_beta_rid_ols:= 0;
        rmse_beta_rid_lav:= 0; sigma_ols:= 0; sigma_rid_ols:= 0; sigma_rid_lav:= 0; cv_ols:= 0;
        cv_rid_ols:= 0; cv_rid_lav:= 0;
        for count:= 1 to iteration do
        begin
          for s:= 1 to sample5 do
            y_ori[s]:=0;
          for s:= 1 to sample5 do
            e[s]:=0;
          mse_ols[count]:= 0;   mse_rid_ols[count]:= 0;   mse_rid_lav[count]:= 0; get_error(n,varian,e);
```

```
  get_beta(p+1,beta); get_y(x_model,beta,e,n,y_ori);

  {---------- OLS ----------}
  x_ols := x_model;   xtxcross(x_ols,n,p+1,xtx_ols);
  xtycross(x_ols,y_ori,n,p,xty_ols);   inv(xtx_ols,p+1,x_inv_ols);
  estimat_beta(x_inv_ols,xty_ols,p,beta_ols);
  rmse_beta_ols := rmse_beta_ols + get_mse_beta(beta_ols,p+1);

  {---------- RID-OLS ----------}
  s_ols:= get_var(x_ols,y_ori,beta_ols,n,p+1);  k_ols:= get_k(p+1,s_ols,beta_ols);
  xtxplusk(xtx_ols,p+1,k_ols,xtx_rid_ols);  inv(xtx_rid_ols,p+1,x_inv_rid_ols);
  estimat_beta(x_inv_rid_ols,xty_ols,p,beta_rid_ols);
  rmse_beta_rid_ols := rmse_beta_rid_ols + get_mse_beta(beta_rid_ols,p+1);

  {---------- RID-LAV ----------}
  leas_abs(x_model,y_ori,n,p,beta_lav);  s_lav:=get_var(x_model,y_ori,beta_lav,n,p+1);
  k_rid_lav:=get_k(p+1,s_lav,beta_lav);   xtxplusk(xtx_ols,p+1,k_rid_lav,xtx_rid_lav);
  inv(xtx_rid_lav,p+1,x_inv_rid_lav);  estimat_beta(x_inv_rid_lav,xty_ols,p,beta_rid_lav);
  rmse_beta_rid_lav:= rmse_beta_rid_lav + get_mse_beta(beta_rid_lav,p+1);
  mse_ols[count]:= get_mse_beta(beta_ols,p+1);
  mse_rid_ols[count]:= get_mse_beta(beta_rid_ols,p+1);
  mse_rid_lav[count]:= get_mse_beta(beta_rid_lav,p+1);
end;
amse_beta_ols := rmse_beta_ols/iteration;  amse_beta_rid_ols := rmse_beta_rid_ols/iteration;
amse_beta_rid_lav := rmse_beta_rid_lav/iteration;
for count:= 1 to iteration do
begin
  sigma_ols:= sigma_ols + power((mse_ols[count]-amse_beta_ols),2);
  sigma_rid_ols:= sigma_rid_ols + power((mse_rid_ols[count]-amse_beta_rid_ols),2);
  sigma_rid_lav:= sigma_rid_lav + power((mse_rid_lav[count]-amse_beta_rid_lav),2);
end;
sigma_ols:= sqrt(sigma_ols/iteration);  sigma_rid_ols:= sqrt(sigma_rid_ols/iteration);
sigma_rid_lav:= sqrt(sigma_rid_lav/iteration);
cv_ols:= sigma_ols/amse_beta_ols;  cv_rid_ols:= sigma_rid_ols/amse_beta_rid_ols;
cv_rid_lav:= sigma_rid_lav/amse_beta_rid_lav;
writeln('  amse beta');
writeln('  amse beta ols = ',amse_beta_ols:30:3);  writeln('  amse beta rid ols = ',amse_beta_rid_ols:30:3);
writeln('  amse beta rid lav = ',amse_beta_rid_lav:30:3);
writeln;
writeln('  standard  deviation');  writeln('  sigma ols = ',sigma_ols:30:3);
writeln('  sigma rid ols = ',sigma_rid_ols:30:3);  writeln('  sigma rid lav = ',sigma_rid_lav:30:3);
```

```
        writeln;
        writeln('  coefficient variance');  writeln('  coefficient variance ols = ',cv_ols:30:3);
        writeln('  coefficient variance rid ols = ',cv_rid_ols:30:3);
        writeln('  coefficient variance rid lav = ',cv_rid_lav:30:3);
        writeln;
        if(amse_beta_ols < amse_beta_rid_ols)then
          amse_min:= amse_beta_ols
        else
          amse_min:= amse_beta_rid_ols;
        if(amse_beta_rid_lav < amse_min)then
          amse_min:= amse_beta_rid_lav;
        writeln('  amse min = ',amse_min:30:3);
        if(amse_min = amse_beta_ols)then
        begin
          writeln('  dif rid ols = ',((amse_beta_rid_ols-amse_min)/amse_min):30:3);
          writeln('  dif rid lav = ',((amse_beta_rid_lav-amse_min)/amse_min):30:3);
        end
        else
          if(amse_min=amse_beta_rid_ols)then
          begin
            writeln('  dif ols = ',((amse_beta_ols-amse_min)/amse_min):30:3);
            writeln('  dif rid lav = ',((amse_beta_rid_lav-amse_min)/amse_min):30:3);
          end
          else
          begin
            writeln('  dif ols = ',((amse_beta_ols-amse_min)/amse_min):30:3);
            writeln('  dif rid ols = ',((amse_beta_rid_ols-amse_min)/amse_min):30:3);
          end;
         writeln;
       end;
       readln;
     end;
  end;
  readln;
end.
```

ประวัติผู้เขียนวิทยานิพนธ์

นายสมลักษณ์ ศิริชื่นวิจิตร เกิดเมื่อวันที่ 31 มีนาคม พ.ศ.2519 สำเร็จการศึกษาปริญญาวิทยาศาสตรบัณฑิต(วท.บ.)สาขาคณิตศาสตร์ คณะวิทยาศาสตร์ มหาวิทยาลัยนเรศวร ในปีการศึกษา 2542 และเข้าศึกษาต่อในหลักสูตรสถิติศาสตรมหาบัณฑิต ภาควิชาสถิติ คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2546