8 REFERENCES

1. Bloise,R, 'La modelisation: objectifs, contributions et limites dans le domaine du traitement des minerais', Industrie Minerale - Les Techniques, v84, no9, November 1984, pp677-680.

2.Brereton,T, 'Performance measurement of a screening process using statistical techniques',The Quarry Managers' Journal, September 1970, pp344-346.

3.Brereton,T & Dymott,KR, 'Some factors which influence screen performance',Proc. 10th Int. Min. Process. Cong., London, 1973, pp181-194.

4.Broussard,A & Albera,F, 'Selection of a model for the simulation of a pilot plant rod mill',Preprint for 5th IFAC Symopsium on Control in Mining, Mineral and Metal Industries, Tokyo, 1986.

5.Caceci,MS & Cacheris,WP, 'Fitting Curves to Data', Byte, May 1984, pp340-362.

6.Calanog,EM & Geiger,GH, 'How to optimize crushing and screening through computer-aided design',E/MJ, May 1983.

7.Cilliers,JJ, Masters thesis in preparation, University of the Witwatersrand, 1987.

8.Cilliers,JJ, Stange,W & King,RP, Microsim Version 2.0 Users guide,University of the Witwatersrand, 1986.

9.Cunningham,CVB, 'Rock Fragmentation Related to Blast Design', The Institute of Quarrying, South Africa, 1982.

10.English,JE, 'A new approach to the theoretical treatment of the mechanics of sieving and screening', Filtration & Separation, March 1974.

11.Evans,LB, Foundations of computer-aided chemical process design,v1,pp425-468, New York, 1980.

12.Ferrara,G & Preti,U, ' A contribution to screening kinetics',Proc 11th IMPC, Cagliari,1975.

13.Ferrara,G, Preti,U & Schena,GD, Modelling of screening operations, University of Trieste, 1986.

14.Ferrara,G, Preti,U & Schena,GD, 'Computer-aided Use of a Screening Process Model',APCOM 87, SAIMM, Johannesburg, 1987, pp153-166.

15.Flavel,MD & Rimmer,HW, Particle breakage studies in an impact crushing environment, Soc of Mining Eng, AIME, 1981.

16. Ford,MA, Simulation of ore dressing plants, PhD thesis, University of the Witwatersrand, 1979.

17.Ford,MA & King,RP, 'The Simulation of ore dressing plants',Int J Min Proc.,v12,1984,pp285-304.

18.Gaudin,AM, Principles of Mineral Dressing, McGraw-Hill,1939.

19. Gottfried,BS & Tierney,JW, 'How to use your microcomputer to simulate a coal preparation plant', Coal Age,november 1985,pp83-84.

20.Gottfried,BS & Tierney,JW, 'Simulation of coal preparation plants', Simulation,v47:4,1986,pp137-144.

21.Hayden,J, The modelling and optimisation of aggregate crushing plants, Continuing Engineering Education course notes, University of the Witwatersrand, 1986.

22. Herbst,JA, Schena,GD & Fu,LS, 'Computerized design of comminution circuits' SME-AIME annual meeting,1986.

23. Hess,FW, 'Interactive colour graphics process analyzer and simulator for mineral concentrators', Preprint for 18th Int APCOM Symp.,London,1984.

24.Hodouin,D, Berube,MA & Everell,MD, 'Modelling industrial grinding circuits and applications in design',CIM Bulletin, September 1978,pp138-146.

25.Karra,VK, 'Calculating the circulating load in crushing circuits', E/MJ, February 1979, pp114-116.

26.Karra, VK, 'A process performance model for cone crushers', Proc. 14th IMPC, Toronto, 1982,III6.1-III6.14.

27. Karra,VK, 'Development of a model for predicting the screening performance of a vibrating screen', CIM Bulletin,April 1979,pp167-171.

28.Karra,VK & Magerowski,AJ, 'Computer-aided crushing circuit design', SA Mining, Coal, Gold & Base Metals, April 1985,pp35-49.

29.Kelly,EG & Spottiswood, DJ, Introduction to mineral processing, John Wiley & Sons,New York,1982.

30. King, EH, 'How to determine plant screening requirements', CEP,May 1977,pp74-79.

31.King,RP, A user's guide to Modsim, University of the Witwatersrand,1986.

32.King,RP, Modsim, University of the Witwatersrand,1986.

33.King,RP, 'Minerals processing - computers can help', SA Mining, Coal, Gold and Base Metals, January 1985.

34.King,RP & Stange,W, Microsim v2.0 Technical reference, University of the Witwatersrand, 1986.

35.Laguitton,D, 'Methodology transfer for the simulation of mineral and coal processing plants', CIM Bulletin,April 1982,pp166-170.

36.Leonard,JW, Determining screen efficiency.

37.Lynch,AJ, Mineral crushing and grinding circuits - their simulation, optimisation, design and control, Elsevier, Amsterdam,1977.

38.Nordberg Process Machinery Manual and appendices, 1976.

39.Nordberg Gyradisc Crusher Manual.

40.Partridge,AC & Roberts, J, 'Principles of screening',Pit & Quarry, December 1977, pp33-38.

41.Pederson, WS & Gurun, T, 'Unit operation design programs', Mining Congress Journal, November 1970, pp60-66.

42.Perry,RH & Green,D eds, Perry's chemical engineers' handbook,6ed, McGraw-Hill, New York,1984.

43.Petno, S & Tompos, E, 'About the new index numbers of separation', Acta Technica Academiae Scientiarum Hungaricae, Tomus, v78, no 1-2, pp237-256.

44.Plitt, LR, 'A mathematical model for the hydrocyclone classifier',CIM Bulletin, December 1986, pp114-123.

45.Richardson, JM, Coles, DR & White, JW, 'Flexmet a computer-aided and flexible metallurgical technique for steady-state flowsheet analysis',E&MJ, October 1981, pp88-97.

46.Ritchie, IC & Spencer, R, 'Economic evaluation of minerals extraction processes by use of a flexible process simulation program', 18th Int. APCOM Symp., London,1984, pp211-223.

47.Robert, M ao, 'Theoretical models of, and models representing, the size reduction processes in ore treatment', Industrie Minerale - Les Techniques, no 9-84, November 1984, pp681-690.

48.Rogers, RSC, 'A classification function for vibrating screens', Powder Technology,v31, 1982, pp135-137.

49.Rose,HE, ' Mechanics of sieving and screening',Trans Inst Min & Metall, v86,ppc101-c114.

50.Stange,W, Cilliers,JJ & King, RP, Microsim Version 3 Technical Reference Manual, University of the Witwatersrand, 1988.

51.Stange,W, Cilliers,JJ & King, RP, Microsim Version 3 User Manual, University of the Witwatersrand, 1988.

52.Sullivan, JF, Screening technology handbook, Triple/S Dynamics, Inc,1975.

53.Taggart, AF, Handbook of Mineral dressing, Wiley,1945.

54.Trelleborg Screening brochure.

55.Walter, GW & Whiten WJ, 'An examination of tertiary screening using simulation', Proc Australasian Inst Min Metall. No 261, March 1977,pp13-16.

56.Whiten, WJ, 'The simulation of crushing plants with models developed using multiple spline regression',10th Int APCOM Symp,,J S Afr Inst Min Metall, May 1972, pp257-264.

57.Whiten, WJ, Walter, GW & White, ME, ' A breakage function suitable for crusher models',4th Tewksbury Symp, Melbourne, February, 1979, pp19.1-19.32.

58.Wills, BA, Mineral Processing Technology, 3ed, Pergamon Press,Oxford,1985.

APPENDIX 1: SOFTWARE, APPARATUS AND TESTS USED

1 Software

After investigating the ore dressing simulators available, as documented in section 3, it was decided to use Modsim, available from the University of the Witwatersrand. The first version of the package was obtained in February 1986. However, the package was not entirely debugged, and two further releases were obtained before September 1986.

Since extensive work had been done on Microsim by the University of the Witwatersrand in the meantime, it was decided to use Microsim rather than Modsim, because of Microsim's expandability to hydrometallurgical stream definitions and because of its superior structure. These features have been documented in section 3.

Microsim Version 1.0 was obtained in September 1986. However, this too was not completely debugged, and was superceded by Microsim Version 2.0 in November 1986. Microsim Version 3.0, with optimisation and design capabilities, was used from January 1989.

Models added to the package for this plant were

CRUSH5
CRUSH6_7 (replaced CRUSH2_3)
SCREEN5
ROSE1
ROSE2, as described in section 4.

In order to incorporate new models into Microsim, additional program modifications have to be made to files TYPEDEF.PAS, MODEL.DAT, and MODELS.PAS. These modifications will not be given. They serve the purpose of directing the simulator's executive program to the correct model and to allocate the correct model storage space. Furthermore, the simulator's database of form-filling input screens and default parameters must be updated. This is done through compiling and executing DBASE and modifying the file MICROSIM.MOD.

Also necessary for the running of Microsim and the incorporation of

new models are DOS Version 3 (or higher) and the TURBO Pascal Version 3.0 compiler.

2 Apparatus

All software was run on an IBM-AT microcomputer with 1MB memory, 1 floppy disk drive and a 40MB hard disk. An IBM EGA graphics screen was used for the flowsheet input and a Roland DXY-980A plotter for output of flowsheets, partition curves and size distributions. The printer used was a Fujitsu DX2200 printer.

3 Tests

At the iron ore mine the following tests were performed:

- -screen analyses of material
- -chemical analyses of ore for grade

At a pilot plant in Pretoria, impact work index tests were done.

4 Data gathered

Data gathered for this work were from records at the iron ore beneficiation plant.

Information on Nordberg crushers was obtained directly from Nordberg.

Information on Microsim was obtained from the Metallurgy department of the University of the Witwatersrand.

APPENDIX 2: CORRELATIONS FOR THE KARRA MODELS

The correlations for factors A through F for the Karra model (27), and A through H for the enhanced model are calculated as follows:

Once the following have been determined:

Q=percentage oversize in feed
R=percentage half size in feed
T=theoretical undersize flow rate in tph
U=bulk density in kg/m^3

and the following parameters are known for one or two decks:

screen area in m^2, S
angle of inclination, theta
square mesh size in mm, w
length of aperture in mm, l
wire (or poly) thickness in mm, d
poly or wire deck

for one deck at a time, the effective aperture, h, in mm, and the factors A through F may be calculated from the empirical correlations of Karra and Nordberg (27,38):

$h=(w+d)\cos(\text{theta}) - d$

factor A:

if h<50,8 $A=12,1286h^{0,3162}-10,2991$
else $A=0,3388h+14,4122$

factor B:

if Q>87, $B=-0,012Q+1,6$
else $B=0,0425Q+4,275$

factor C:

if $R \leq 30$, $C=0,012R+0,7$
if $30<R<55$, $C=0,1528R^{0,564}$
if $55 \leq R<80$, $C=0,0061R^{1,37}$
if $R \geq 80$, $C=0,05R-1,5$

```
factor D:
        if top deck, D=1,0
        if second deck, D=0,9

factor E:
        let t=1,26h
        if t<1,      E=1,0
        if 1≤t≤2,    E=t
        if 2<t<4,    E=1,5+0,25t
        if 4≤t≤6,    E=2,5
        if 6<t≤10,   E=3,25-0,125t
        if 10<t<12,  E=4,5-0,25t
        if 12≤t≤16,  E=2,1-0,05t
        if 16<t<24,  E=1,5-0,125t
        if 24≤t≤32,  E=1,35-0,00625t
        if t>32,     E=1,15

factor F:
        F=U/1602

factor G (determined from poly decks used):
        if wire deck, G=1,0
        if poly deck, G=1/1,15

factor H (given in Kelly and Spottiswood (29):
        if l/w<2,    H=1,0
        if 2≤l/w<3,  H=1,1
        if 3≤l/w<6,  H=1,4
        if l/w≥6,    H=1,6
```

APPENDIX 3: PASCAL LISTINGS OF MODELS

The Pascal listings for new and modified Microsim models are given below. Models were written in accordance with the Microsim data structure as given in the Technical Reference Manual (50).

The crushing models CRUSH5 and CRUSH6_7 are contained in the Microsim library file UNIT1.PAS CRUSH6_7 replaces the general Whiten model CRUSH2_3, of which it is a modification for haematite, whereas CRUSH5 is a new model for gyradisc crushing of haematite.

The screening models ROSE1, ROSE2 and SCREEN5 were added to the Microsim library file UNIT2.PAS. SCREEN5 is an enhancement of the original Karra model, SCREEN4. ROSE1 and ROSE2 are both new models.

A3.1 Gyradisc model CRUSH5

```
{-------------------------------------------------------------------------}
{  GYRADISC MODEL - Sishen test model                                      }
{-------------------------------------------------------------------------}
OVERLAY PROCEDURE CRUSH5( var stm          : stms  ;
                          point            : tounit;
                          type_num         : UNITYPES);

   VAR
   feedpt,prodpt,der2pt                : tor4;
   feedh2o,prodh2o                     : tor6;

   g,nd,yl,y,ws,wsl,ak,an              : real ;

   i,j,k,l,m                           : integer ;

   BEGIN
     feedpt := solidselect(stm[point^.inout[2]]) ;
     prodpt := solidselect(stm[point^.inout[4]]) ;

     feedh2o := lixivselect(stm[point^.inout[2]]) ;
     prodh2o := lixivselect(stm[point^.inout[4]]) ;
     MAKESCRATCH(der2pt);

     prodh2o^.flow  := feedh2o^.flow  ;
     prodpt^.tonnes := 0.0 ;

     g := point^.param[1];

     WITH system DO
     begin
      FOR j:=1 TO ngc DO
       FOR k:=1 TO nsc DO
        FOR l:=1 TO n4c DO
         FOR m:=1 TO n5c DO
          SOLIDPUT(1,j,k,l,m,der2pt,0.0);

      FOR i:=1 TO ndc DO
       FOR j:=1 TO ngc DO
        FOR k:=1 TO nsc DO
         FOR l:=1 TO n4c DO
          FOR m:=1 TO n5c DO
           SOLIDPUT(1,j,k,l,m,der2pt,SOLIDVALUE(i,j,k,l,m,feedpt)+
                                     SOLIDVALUE(1,j,k,l,m,der2pt));
```

```
      nd := ndc -1 ;                          y1 := 1.0     ;
      FOR i := 1 to ndc DO
      BEGIN  (* for i *)
        ws := 0.0 ;
        IF i < ndc THEN ws := sqrt(size[i]*size[i+1])/g;

        IF ws >= 0.15  THEN
         begin
          an := 2.77 ;
          ak := 0.38 ;
         end
        ELSE
          begin
           an := 0.32 ;
           ak := 250.0 ;
          end;

        ws1 := POW((ws/ak),an);
        y := 1.0 ;
        IF ws1 < 10 THEN y := 1.0 - EXP(-ws1);

        FOR j := 1 TO ngc DO
         FOR k := 1 to nsc DO
          FOR l := 1 to n4c DO
           FOR m := 1 to n5c DO
            SOLIDPUT(i,j,k,l,m,prodpt,(y1-y)*SOLIDVALUE(1,j,k,l,m,der2pt));
        y1:=y;
      END; {For i}

      FOR i:=1 TO ndc DO
       FOR j:=1 TO ngc DO
        FOR k:=1 TO nsc DO
         FOR l:=1 TO n4c DO
          FOR m:=1 TO n5c DO
           prodpt^.tonnes := prodpt^.tonnes + SOLIDVALUE(i,j,k,l,m,prodpt) ;

        KILLSCRATCH(der2pt);
       END ;(* with system *)
    END; (* crush5 *)
-------------------------------------------------------------------------------
```

A3.2 Whiten model CRUSH6_7

```
{-------------------------------------------------------------------------}
{  CONE CRUSHER MODELS 6 and 7  : only for haemetite                       }
{          Standard Cone crusher model : Based on Whitens model           }
{          Short Head crusher model    : Based on Whitens model           }
{-------------------------------------------------------------------------}
OVERLAY PROCEDURE CRUSH6_7( var stm          : stms   ;
                            point            : tounit ;
                            type_num         : UNITYPES);

    VAR
    feedpt,prodpt,out2pt              : tor4;
    feedh2o,prodh2o                   : tor6;
    c,ws,ws1,w                        : REAL;
    b,b1,bii                          : REAL;
    i,j,k,l,m,ll,l1                   : integer ;


PROCEDURE WHITEN(css,kk,k1,k2,k3,n,mm : REAL);
                                       {Common calculation part}
BEGIN
 WITH system DO
 BEGIN
  FOR i:=1 TO ndc DO
  BEGIN
   c:=1;
   IF (size[i]<k2) THEN
   c:=1-POW((size[i]-k2)/(k1-k2),k3);
   IF (size[i]<k1) THEN
    c:=0.0;
   ws1:=0.0;                          ws:=size[1];
   IF (i<ndc) THEN
    ws1:=SQRT(size[i]*size[i+1]);
   IF (i>1) THEN
    ws:=SQRT(size[i-1]*size[i]);

   FOR j:=1 TO ngc DO
    FOR k:=1 TO nsc DO
     FOR l:=1 TO n4c DO
      FOR m:=1 TO n5c DO
       BEGIN
        w:=SOLIDVALUE(i,j,k,l,m,feedpt);            ll:=i-1;
        IF (ll<>0) THEN
        BEGIN
         FOR ll:=1 TO ll DO
         BEGIN
          b1:=(1-kk)*POW(ws1/size[ll],n)+kk*POW(ws1/size[ll],mm);
          b:=(1-kk)*POW(ws/size[ll],n)+kk*POW(ws/size[ll],mm)-b1;
          w:=w+b*SOLIDVALUE(ll,j,k,l,m,out2pt);
         END; {For l}
        END; {If ll}
        bii:=1-(1-kk)*POW(ws1/size[i],n)-kk*POW(ws1/size[i],mm);
        w:=w/(1-bii*c);                  SOLIDPUT(i,j,k,l,m,out2pt,w*c);
        SOLIDPUT(i,j,k,l,m,prodpt,w*(1-c));
        prodpt^.tonnes := prodpt^.tonnes+SOLIDVALUE(i,j,k,l,m,prodpt) ;
   END; {j,k,l,m}
  END; {For I}
 END; {with system}
END; {Calculate procedure}
{-------------------------------------------------------------------------}
VAR
  css,kk,k1,k2,k3,n,mm            : REAL;


BEGIN
 feedpt := solidselect(stm[point^.inout[2]]) ;
 prodpt := solidselect(stm[point^.inout[4]]) ;
 feedh2o := lixivselect(stm[point^.inout[2]]) ;
 prodh2o := lixivselect(stm[point^.inout[4]]) ;
```

```
 MAKESCRATCH(out2pt);

 prodh2o^.flow  := feedh2o^.flow  ;
 prodpt^.tonnes := 0.0 ;

 css:=point^.param[1];                kk:=point^.param[2];

 CASE point^.MODEL OF
  2 : BEGIN
       k1:=0.653*css;     k2:=1.21*css;     k3:=2.0;
       n:=2.0;    mm:=0.535;
      END; {Standard Symons cone crusher for haemetite}
  3 : BEGIN
       k1:=0.944*css;     k2:=1.722*css+0.004826;        k3:=3.0;
       n:=2.000;          mm:=0.535;
      END; {Short head cone crusher for haemetite}
 END;
 WHITEN(css,kk,k1,k2,k3,n,mm);
 KILLSCRATCH(out2pt);
END; {Procedure}
```

A3.3 Single deck Rose efficiency model ROSE1

```
{-------------------------------------------------------------------------}
{ SINGLE DECK ROSE MODEL                                                  }
{-------------------------------------------------------------------------}
OVERLAY PROCEDURE ROSE1 ( var stm          : stms    ;
                            point          : tounit ;
                            type_num       : UNITYPES);
VAR
 feedpt,out1pt,out2pt              : tor4;
 feedh2o,out1h2o,out2h2o           : tor6;
 eff,value,value2,d50,x,pof        : real ;
 i,j,k,l,m                         : integer ;

BEGIN
 feedpt := solidselect(stm[point^.inout[2]]) ;
 out1pt := solidselect(stm[point^.inout[4]]) ;
 out2pt := solidselect(stm[point^.inout[5]]) ;
 feedh2o := lixivselect(stm[point^.inout[2]]) ;
 out1h2o := lixivselect(stm[point^.inout[4]]) ;
 out2h2o := lixivselect(stm[point^.inout[5]]) ;
 out1pt^.tonnes := 0.0 ;
 out2pt^.tonnes := 0.0 ;
 eff            := point^.param[2];
 d50            :=eff*point^.param[1];

 WITH system DO
 BEGIN
  FOR i := 1 to ndc DO
    BEGIN
     x:=POW(size[i]/d50,5.846);
     IF (x<60) THEN pof:=1-EXP(-0.693*x)
     ELSE pof:=1;
     IF pof>1 THEN pof:=1
     ELSE IF pof<0 THEN pof:=0;
   FOR j := 1 TO ngc DO
    FOR k := 1 to nsc DO
     FOR l := 1 to n4c DO
      FOR m := 1 to n5c DO
      BEGIN
       value := 0.0 ;
       IF size[i] <= point^.param[1] THEN
        value := (1-pof)* solidvalue(i,j,k,l,m,feedpt);
       value2 := solidvalue(i,j,k,l,m,feedpt) - value;

       SOLIDPUT(i,j,k,l,m,out1pt,value);
       SOLIDPUT(i,j,k,l,m,out2pt,value2);

       out1pt^.tonnes := out1pt^.tonnes + value ;

       out2pt^.tonnes := out2pt^.tonnes + value2;
       END;
      END; (* for *)
 END; (* with system *)
 out1h2o^.flow := 0.95*feedh2o^.flow;
 out2h2o^.flow := 0.05*feedh2o^.flow;
END; (* rose1 *)
```

A3.4 Double deck Rose efficiency model ROSE2

```
{-------------------------------------------------------------------------}
{ DOUBLE DECK ROSE MODEL                                                  }
{-------------------------------------------------------------------------}
OVERLAY PROCEDURE ROSE2 ( var stm       : stms   ;
                          point         : tounit ;
                          type_num      : UNITYPES);

    VAR
    feedpt,outlpt,out2pt,out3pt       : tor4;
    feedh2o,outlh2o,out2h2o,out3h2o   : tor6;

    eff1,eff2,d501,d502,x1,x2,pof1,pof2  : real ;
    value,value1,value2 ,value3          : real ;

    i,j,k,l,m .                          : integer ;


    BEGIN

      feedpt := solidselect(stm[point^.inout[2]]) ;
      outlpt := solidselect(stm[point^.inout[4]]) ;
      out2pt := solidselect(stm[point^.inout[5]]) ;
      out3pt := solidselect(stm[point^.inout[6]]) ;

      feedh2o := lixivselect(stm[point^.inout[2]]) ;
      outlh2o := lixivselect(stm[point^.inout[4]]) ;
      out2h2o := lixivselect(stm[point^.inout[5]]) ;
      out3h2o := lixivselect(stm[point^.inout[6]]) ;

      outlpt^.tonnes := 0.0 ;
      out2pt^.tonnes := 0.0 ;
      out3pt^.tonnes := 0.0 ;

      eff1            := point^.param[2];
      eff2            := point^.param[4];
                      d501:=eff1*point^.param[1];
                      d502:=eff2*point^.param[3];

      WITH system DO begin

        FOR i := 1 to ndc DO
        BEGIN
         x1:=POW(size[i]/d501,5.846);
         x2:=POW(size[i]/d502,5.846);
        IF (x1<60) THEN pof1:=1-EXP(-0.693*x1)
        ELSE pof1:=1;
        IF pof1>1 THEN pof1:=1
        ELSE IF pof1<0 THEN pof1:=0;
        IF (x2<60) THEN pof2:=1-EXP(-0.693*x2)
        ELSE pof2:=1;
        IF pof2>1 THEN pof2:=1
        ELSE IF pof2<0 THEN pof2:=0;
        FOR j := 1 TO ngc DO
        FOR k := 1 to nsc DO
        FOR l := 1 to n4c DO
        FOR m := 1 to n5c DO
         BEGIN
          value := solidvalue(i,j,k,l,m,feedpt) ;
          IF size[i] < point^.param[1] THEN
                       value2 := (pof1) * value  ;
          IF size[i] >= point^.param[1] THEN
                       value2 := value;
          IF size[i] < point^.param[3] THEN
                       value3 := (pof2)*(value-value2);
          IF size[i] >= point^.param[3] THEN
                       value3 := (value-value2);

         value1 := value-value2-value3;
```

```
        solidput(i,j,k,l,m,out1pt,value1);
        solidput(i,j,k,l,m,out2pt,value2);
        solidput(i,j,k,l,m,out3pt,value3);

        out1pt^.tonnes := out1pt^.tonnes + value1;
        out2pt^.tonnes := out2pt^.tonnes + value2;
        out3pt^.tonnes := out3pt^.tonnes + value3;
                                  END;
        END; (* for *)


      out1h2o^.flow := feedh2o^.flow * 0.9 ;
      out2h2o^.flow := feedh2o^.flow * 0.05;
      out3h2o^.flow := feedh2o^.flow * 0.05;
    END; (* with system *)
  END; (* rose2 *)
```

A3.5 Enhanced Karra Model SCREEN5

```
{-------------------------------------------------------------------------}
{ SCREEN MODEL 5                                                          }
{         Enhanced  Karra screen model for single or double screens       }
{-------------------------------------------------------------------------}
OVERLAY PROCEDURE SCREEN5(VAR stm : STMS;
                             point : TOUNIT;
                         type_num : UNITYPES);


 PROCEDURE PARTICLE_SIZE(stream_ptr : TOR4;
                     VAR cumulative : SIZES);

 VAR
  cumulat : REAL;
  i       : INTEGER;

 BEGIN
  cumulat:=0;
  FOR i:=1 TO system.NDC DO
  BEGIN
   cumulat:=cumulat+SIGNDC(i,stream_ptr);
   cumulative[i]:=100-cumulat;
  END;
 END;
 {-------------------------------------------------------------------------}
 FUNCTION INTERPOLATE( x,y : SIZES;
                         n : INTEGER;
                        xx : REAL) : REAL;

 VAR
  c                        : ARRAY[sizrange,1..3] OF REAL;
  i,n0,k                   : INTEGER;
  rm3,t1,rm2,rm1,rm4,t2,b  : REAL;

 BEGIN
  rm3:=(y[2]-y[1])/(x[2]-x[1]);          t1:=rm3-(y[2]-y[3])/(x[2]-x[3]);
  rm2:=rm3+t1;                           rm1:=rm2+t1;
  n0:=n-2;

  FOR i:=1 TO N DO BEGIN
   IF (i>n0) THEN
    rm4:=rm3-rm2+rm3
   ELSE
    rm4:=(y[I+2]-y[i+1])/(x[I+2]-x[I+1]);
   t1:=ABS(rm4-rm3);                    t2:=ABS(rm2-rm1);
   b:=t1+t2;
   IF (b<>0) THEN
    c[i,1]:=(t1*rm2+t2*rm3)/B
   ELSE
    c[i,1]:=0.5*(rm2+rm3);
   rm1:=rm2;      rm2:=rm3;          rm3:=rm4
  END;

  n0:=n-1;

  FOR i:=1 TO n0 DO BEGIN
   t1:=1/(x[I+1]-x[i]);                 t2:=(y[i+1]-y[i])*t1;
   b:=(c[i,1]+c[i+1,1]-t2-t2)*t1;       c[i,3]:=b*t1;
   c[i,2]:=-b+(t2-c[i,1])*t1
  END;

  FOR i:=1 TO n-1 DO
   IF ((xx>=x[i]) AND (xx<x[I+1])) THEN k:=i;
  xx:=xx-x[k];
  interpolate:=c[k,3]*xx*xx*xx+c[k,2]*xx*xx+c[k,1]*xx+y[k]
 END;
```

```
{-------------------------------------------------------------------------------}
 FUNCTION  PERCENTAGE_PASSING(cumulative  :  SIZES;          {Find  %  passing psize}
                              psize       :  REAL) : REAL;   {psize  in  microns    }
 VAR
  x,y          : SIZES;
  i            : INTEGER;
  dummy        : REAL;

 BEGIN
  psize:=psize/1E6;                    {Convert to metres}
  WITH system DO
  BEGIN
   FOR i:=1 TO ndc DO
   BEGIN
    x[i]:=size_class[ndc-i+1];
    y[i]:=cumulative[ndc-i+1];
   END;
   IF psize>size_class[1] THEN
   BEGIN
    dummy:=psize/size_class[1]*cumulative[1];
    IF dummy>100 THEN dummy:=100;
   END
   ELSE
    dummy:=INTERPOLATE(x,y,ndc,psize);
  END;
  percentage_passing:=dummy;
 END;
  {-------------------------------------------------------------------------}
  PROCEDURE MODCALC_ONE_DECK(feedpt,ofpt,ufpt : TOR4;
                             lw,ht,area,poly : REAL;              {Ht in mm}
                                       deck  : INTEGER;
                                       water : REAL);

  VAR
   i,j,k,l,m                                    : INTEGER;
   a,b,c,d,e,f,kk,xn,p,q,r,t,g,h                : REAL;
   undersize_theory,svm,sga,u,pof,d50,x         : REAL;
   cumulative                                   : SIZES;

  BEGIN
   PARTICLE_SIZE(feedpt,cumulative);                  {Get cumulative in feed}
   q:=100-PERCENTAGE_PASSING(cumulative,ht*1000);     {% oversize in feed    }
   r:=PERCENTAGE_PASSING(cumulative,ht*500);          {% half size in feed   }
   xn:=PERCENTAGE_PASSING(cumulative,1250*ht)-
       PERCENTAGE_PASSING(cumulative,750*ht);         {% near size in feed   }
   undersize_theory:=(1-q/100)*feedpt^.TONNES*3.6;    {Theoretical underflow TPH}
   p:=ht;
   t:=1.26*ht;
                    {Get constants A,B,C,D,E,F,G,H}
   IF (p>=50.8) THEN a:=14.4122+0.3388*p
   ELSE a:=12.1286*POW(p,0.3162)-10.2991;

   IF (q>87) THEN b:=4.275+0.0425*q
   ELSE b:=1.6-0.012*q;

   IF (r>=80) THEN c:=0.05*r-1.5
   ELSE
    IF ((r<80) AND (r>=55)) THEN c:=0.0061*POW(r,1.37)
    ELSE
     IF ((r<55) AND (r>30)) THEN c:=0.1528*POW(r,0.564)
     ELSE c:=0.012*r+0.7;

   d:=1.1-0.1*deck;
   {g = polydeck factor for reduced area}
   {h = factor for nonsquare apertures}
   IF poly=1.0 THEN g:=1./1.15
   ELSE g:=1.0;
   IF lw<2.0 THEN h:=1.0
   ELSE
     IF lw<3.0 THEN h:=1.1
```

```
    ELSE
      IF lw<6 THEN h:=1.4
      ELSE h:=1.6;
  IF (water>0) THEN      {Wet screening}
  BEGIN
   IF (t>32) THEN e:=1.15
   ELSE
    IF ((t<=32) AND (t>=24)) THEN e:=1.35-0.00625*t
    ELSE
     IF ((t<24) AND (t>16)) THEN e:=1.5-0.0125*t
     ELSE
      IF ((t<=16) AND (t>=12)) THEN e:=2.1-0.05*t
      ELSE
       IF ((t<12) AND (t>10)) THEN e:=4.5-0.25*t
       ELSE
        IF ((t<=10) AND (t>6)) THEN e:=3.25-0.125*t
        ELSE
         IF ((t<=6) AND (t>=4)) THEN e:=2.5
         ELSE
          IF ((t<4) AND (t>2)) THEN e:=1.5+0.25*t
          ELSE
           IF ((t<=2) AND (t<=1)) THEN e:=t
           ELSE
            e:=1.0;

   sga:=Ore_Density(feedpt);
   if sga > 0.0 then svm:=1/(1000*sga)
                else svm:=0;
   u:=sga*0.6*1000;
   f:=u/1602;
  END
  ELSE
  BEGIN                                                {Dry screening}
   e:=1;                f:=1;
  END;

  kk:=undersize_theory/area/(a*b*c*d*e*f*g*h);

  d50:=0.975*ht*POW(kk,-0.148)*POW((1-xn/100),0.511);

  d50:=d50*1e-03;                                      {D50 in meters}

  ufpt^.TONNES:=0.0;          ofpt^.TONNES:=0.0;
  WITH system DO
  BEGIN
   FOR i:=1 TO ndc DO
   BEGIN
    x:=POW(size[i]/d50,5.846);
    IF (x<60) THEN pof:=1-EXP(-0.693*x)
    ELSE pof:=1;
    IF pof>1 THEN pof:=1
    ELSE IF pof<0 THEN pof:=0;
    FOR j:=1 TO ngc DO
     FOR k:=1 TO nsc DO
      FOR l:=1 TO n4c DO
       FOR m:=1 TO n5c DO
       BEGIN
        SOLIDPUT(i,j,k,l,m,ofpt,pof*SOLIDVALUE(i,j,k,l,m,feedpt));
        SOLIDPUT(i,j,k,l,m,ufpt,SOLIDVALUE(i,j,k,l,m,feedpt)
                               -SOLIDVALUE(i,j,k,l,m,ofpt));
        ufpt^.TONNES:=ufpt^.TONNES+SOLIDVALUE(i,j,k,l,m,ufpt);
        ofpt^.TONNES:=ofpt^.TONNES+SOLIDVALUE(i,j,k,l,m,ofpt);
       END;
    END;
   END;
  END;
  {-------------------------------------------------------------------------}
VAR
 feedpt,out1pt,out2pt,out3pt,dummyfeedpt : TOR4;
 feedh2o,out1h2o,out2h2o,out3h2o         : TOR6;
```

```
 i,j,k,l,m                                  : INTEGER;
 area,ht,theta,water,lw,poly                : REAL;
 deck                                       : INTEGER;

BEGIN
 feedpt := solidselect(stm[point^.inout[2]]) ;
 outlpt := solidselect(stm[point^.inout[4]]) ;
 out2pt := solidselect(stm[point^.inout[5]]) ;

 IF (type_num=19) THEN
 BEGIN
  out3pt := solidselect(stm[point^.inout[6]]) ;
  out3h2o := lixivselect(stm[point^.inout[6]]) ;
 END;

 feedh2o := lixivselect(stm[point^.inout[2]]) ;
 outlh2o := lixivselect(stm[point^.inout[4]]) ;
 out2h2o := lixivselect(stm[point^.inout[5]]) ;

 MAKESCRATCH(dummyfeedpt);

 area:=point^.param[1];          theta:=point^.param[2]/180*3.1416;
 WITH point^ DO
 BEGIN
  ht:=(param[3]+param[5])*COS(theta)-param[5];
  lw:=param[4]/param[3];
  poly:=param[6];
  END;
 water:=feedh2o^.FLOW*3.6;

 MODCALC_ONE_DECK(feedpt,out2pt,dummyfeedpt,lw,ht,area,poly,1,water);

 WITH system DO
  CASE type_num OF
   15 : BEGIN
         FOR i:=1 TO ndc DO
          FOR j:=1 TO ngc DO
           FOR k:=1 TO nsc DO
            FOR l:=1 TO n4c DO
             FOR m:=1 TO n5c DO
              SOLIDPUT(i,j,k,l,m,outlpt,SOLIDVALUE(i,j,k,l,m,dummyfeedpt));
         outlpt^.TONNES:=dummyfeedpt^.TONNES;
         outlh2o^.FLOW:=0.95*feedh2o^.FLOW;
         out2h2o^.FLOW:=0.05*feedh2o^.FLOW;
        END;
   19 : BEGIN

    WITH point^ DO
    BEGIN
          ht:=(param[7]+param[9])*COS(theta)-param[9];
          lw:=param[8]/param[7];
          poly:=param[10];
          END;
         MODCALC_ONE_DECK(dummyfeedpt,out3pt,outlpt,lw,ht,area,poly,2,water);
         outlh2o^.flow := feedh2o^.flow * 0.9 ;
         out2h2o^.flow := feedh2o^.flow * 0.05;
         out3h2o^.flow := feedh2o^.flow * 0.05;
        END;
  END;

 KILLSCRATCH(dummyfeedpt);
END;
```