Appendices

Appendix A C-codes

This Appendix contains the following programs written in C language:

1) `newnx.c` - reconstructs the flow connectivity between modeling units after some of units have been removed;

2) `fdy4.c`--estimates discharge in ungauged streams.

A1 Program `newnx.c`--reconstructing the flow connectivity between modeling units after some of units have been removed

```
/* newnx.c
* from new1.c, 9/21/95
* Pawel Mizgalewicz, CRWR UT at Austin  */
/*****************************************************************
*  newnx.c -- reconstructs the flow connectivity between units
*  after some of them have been removed. Requires two ASCII, comma
*  delimited files:
*    file 1) full set of units. Each line should contain:
*            unit_id, next unit_id
*    file 2) reduced set of units. Each line should contain the
*            unit_id number
*  Output file: unit_id, id of downstream unit, unit order
*
* USAGE: newnx infile1 infile2 outfile
*****************************************************************/
#include <stdio.h>
main (argc, argv)
  int argc;
  char *argv[];
  /* arg 1) input_file (oid, onx => old ID, old NEXT)
```

```
        2) input_file (nid => new ID)
        3) output_file (nid, nnx, nor => ID, new NEXT, order) */
{
 char loop, found;
 int i, j, k, onn, nnn, nni, nor[1200], nix[1200];
 long id, nx, ix,
      oid[1200], onx[1200], nid[1200], nnx[1200];
 FILE *fgsin, *funin, *ftable2;
 if (argc < 3 )
  {
   printf ("wrong number of arguments = %d \n", argc);
   exit(1);
  }
 fgsin = fopen (argv[1], "r");
 i = 0;
 while (fscanf(fgsin, "%li,%li", &id,&nx) != EOF)
  {
   oid[i] = id;
   onx[i] = nx;
   ++i;
  }
 onn = i-1;
 fclose(fgsin);
 funin = fopen (argv[2], "r");
 i = 0;
 while (fscanf(funin, "%li", &id) != EOF)
  {
   nid[i] = id;
   nnx[i] = 0;
   ++i;
  }
 nnn = i-1;
 nni = i;
 fclose(funin);
 /* To each new unit assign old_next        */
 for (i = 0; i <= nnn; ++i)
   for (k = 0; k <= onn; ++k )
     if ( nid[i] ==  oid[k] )
      {
     nnx[i] = onx[k];
     break;
      }
 /* check if all next are valid (have new_id specified)*/
 for (i = 0; i <= nnn; ++i)
  {
   found = 0;
   ix = 0;
   while (!found)
    {
     ++ix;
     for (j = 0; j <= nnn; ++j)
     if ( ( nnx[i] == nid[j] ) || ( nnx[i] == 0 ) )
      {
       found = 1;
       break;
      }
     if (found) continue;
     /* if not found, go to old table and read next_id  */
     for (j = 0; j <= onn; ++j)
     if ( nnx[i] == oid[j] )
      {
       nnx[i] = onx[j];
       break;
      }
     if(ix > 1000000) printf("error %i\n", ix);
    }
  }
/*  ==============  WATERSHED ORDER ================ */
/* Set initial value of order array */
 for (i = 0; i <= nnn; ++i)
  {
   nor[i] = 1;
   nix[i] = nni;
  }
 /* Select first order items */
 for (i = 0; i <= nnn; ++i)
  {
  j = 0; loop = 1;
  while ( loop )
   {
    if (nnx[i] == nid[j])
     {
      nor[j] = 0;
      nix[i] = j;
      loop = 0;
     }
    ++j;
    if (j == nni) loop = 0;
   }
  }
 /* order of the remaining streams */
 for (i=0; i<=nnn; ++i)
   if(nor[i] == 1)
    {
     j = nix[i];
     k = 2;
     while ((nor[j] < k) && (j != nni))
      {
     nor[j] = k;
     j = nix[j];
     k = k + 1;
      }
    }
 /* Write results to output file */
 ftable2 = fopen (argv[3], "w");
 for (j = 0; j <= nnn; ++j )
   fprintf(ftable2, "%li,%li,%i\n", nid[j], nnx[j], nor[j] );
 fclose(ftable2);
 return 0;
}
```

A2 Program `fdy4.c`--calculating the flow rate in ungauged streams from the available record

```
/**********************************************************/
/* fdy4.c  -- calculates flow rate in all modeling units from
*             the available record. The average precipitation
*             depth is used as a weight.
*  arguments:
*            1) file name that contains USGS flow record
*            2) file name that contains modeling unit data
*            3) name of the output file.
*  input: two ASCII files, each record contains the following
*         values, coma delimited:
*         file one specified by the first argument:
*            gsid = USGS station identification number,
*            gsnx = ID number of the downstream USGS station,
*            gsqo = flow rate;
*         file two specified by the second argument:
*            unid = modeling unit identification number,
*            unnx = ID number of the downstream modeling unit,
*            ungsid = ID number of the USGS station that
*                     determines the zone in which the unit
*                     is located (ID of the gauging station that
*                     is downstream of the modeling unit),
*            unar = area of the modeling unit,
*            unor = modeling unit order in the flow system,
*            unpr = average precipitation depth;
*  output:   ASCII file specified by the third argument:
*            unid = modeling unit identification number,
*            unqc = flow rate estimated in modeling unit
*                   (this flow is cumulative, i.e., it
*                   is a runoff from the drainage area
*                   determined by the modeling unit outlet. */
/**********************************************************/

#include <stdio.h>
#define size 512

main (argc, argv)
  int argc;
  char *argv[];
  /* arg 1) input_file (gs) must have: id, next, and flow.
       2) input (modeling units--un)
          must have: id, next, gsid, area, order, precip.
       3) output_file: unid, qc (cumulative)
      used in ArcView version:
       4) name_id (copied from first column)
       5) name_out */
```

```
 {
  char loop, notfound;
  int i, j, k, l, n, unormx, d;
  int unor[2000], unixx[2000], ungsix[2000], gsused[100], or,
      gsni, gsnn, unni, unnn, unormxi;
  long gsid[100], gsnx[100], id, nx, idgs,
       unid[1200], unnx[1200], ungsid[1200];
  float gsqo[100], gsqi[100], gsvl[100], gscf[100],
      unar[1200], unqo[1200], unpr[1200], unqc[1200], unqct[1200],
      ar, qo, pr, rncf, sum1, sum2, xxx, x2;
  /* gsor[100], gsar[100], gspr[100] deleted, gsvl[100] created */
  char buf[size];
  FILE *fgsin, *funin, *ftable2;
  if (argc < 3 )
   {
    printf ("wrong number of arguments = %d \n", argc);
    exit(1);
   }
  fgsin = fopen (argv[1], "r");
/*  fgets(buf, size, fgsin);  */
  i = 0;
  while (fscanf(fgsin, "%li,%li,%f", &id,&nx,&qo) != EOF)
   {
    gsid[i] = id;
    gsnx[i] = nx;
    gsqo[i] = qo;
    ++i;
   }
  gsnn = i-1;
  gsni = i;
  fclose(fgsin);
/* look for records that has Q = 0. All records that has Q=0 will
   have instead of next-id, the id they should have as a new unit.
*/
    for ( i = 0; i <= gsnn; ++i )
   {
    if ( gsqo[i] > 0.0 ) continue;
    for ( j = 0; j <= gsnn; ++j )
      if ( gsid[i] == gsnx[j]) gsnx[j] = gsnx[i];
   }

/*  =========================================================*/

  funin = fopen (argv[2], "r");
 fgets(buf, size, funin); /* tables unload first, dummy record */
  i = 0;                  /* that have negative area !!!!      */
  while (fscanf(funin, "%li,%li,%li,%f,%i,%f",
       &id,&nx,&idgs,&ar,&or,&pr) != EOF)
   {
    unid[i] = id;
    unnx[i] = nx;
    ungsid[i] = idgs;
    unar[i] = ar;
    unor[i] = or;
    unpr[i] = pr;
    ++i;
   }
  unnn = i-1;
  unni = i;
  fclose(funin);
  /* update ungsid[i] */
  for ( i = 0; i <= gsnn; ++i )
   {
    if ( gsqo[i] > 0.0 ) continue;
    for ( j = 0; j <= unnn; ++j )
      if ( gsid[i] == ungsid[j]) ungsid[j] = gsnx[i];
   }
 /* rebuild GS arrays, i.e. remove records GSQO <= 0  */
    i = 0;
  for ( j = 0; j <= gsnn; ++j )
   {
    if ( gsqo[j] <= 0.0 ) continue;
    gsid[i] = gsid[j];
    gsnx[i] = gsnx[j];
    gsqo[i] = gsqo[j];
    gsvl[i] = 0.0;
    gsqi[i] = 0.0;       /* initialization of the inflow table */
    gsused[i] = 0;
    ++i;
   }
  gsnn = i-1;
  gsni = i;


  /* Set UNGSIX index, i.e. index that relates UNID with the GSID
  = record in UN table with the record in GS table,
    if ungsid[] = 0, i.e.,. no gauging station assigned to this
record
     index = gsnn + 1 = gsnx     */
  for (i = 0; i <= unnn; ++i)
   {
    if ( ungsid[i] == 0 )
     {
      ungsix[i] = gsni;
      continue;
     }
    for (k = 0; k <= gsnn; ++k )
     {
      if ( ungsid[i] ==  gsid[k] )
      ungsix[i] = k;
     }
   }
  for (i = 0; i <= gsnn; ++i )
    {
      for (j = 0; j <= unnn; ++j )
       if ( ungsid[j] == gsid[i] )
       {
        gsvl[i] = gsvl[i] + ( unpr[j] * unar[j] );
      }
    }
  /* Calculate runoff coefficient, use first order GS watersheds only
*/
  sum1 = 0.0;
  sum2 = 0.0;
  xxx = 1.0 / 8640.0 ;  /*  area in km2, prec in 0.001 cm/d, -> m3/s
*/
  notfound = 1;
  for (i = 0; i <= gsnn; ++i)
   {
    notfound = 1;
    for (j = 0; j <= gsnn; ++j )
     {
      if (gsnx[j] == gsid[i] )
      {
        notfound = 0;
        break;
      }
     }
    if ( notfound == 1 )
     {
     ++n;
     sum1 = sum1 + gsqo[i];
     sum2 = sum2 + ( gsvl[i] * xxx);
     }
   }
/*   printf("s1  s2  %f,%f\n",sum1,sum2);  */
   rncf = sum1 / sum2;

  /* 1) Calculate runoff from all UN watersheds */
  /* 2) assign initial values to both, unqc[] and unqct[]  */
  /* 3) fill array unixx[], index of the next/downstream unit */
  /*    is set in "find maximum order" module */

  for (i = 0; i <= unnn; ++i)
   {
    unqo[i] = rncf * unpr[i] * unar[i] * xxx ;
    unqc[i] = unqo[i];
    unqct[i] = unqo[i];
    for ( j = 0; j <= unnn; ++j)
     {
      if (unid[j] == unnx[i])
      {
       unixx[i] = j;
       break;
      }
     }
   }

  /* Find maximum UN order */
  unormx = 0;
  for (i = 0; i <= unnn; ++i)
   {
    if ( unormx < unor[i] )
     {
      unormx = unor[i];
      unormxi = i;
     }
   }
   unixx[unormxi] = unni;
  /* Calculate zonal cumulative UN flow (zonal = within GS zone) */
  for (k = 1; k < unormx; ++k)
   {
    for (i = 0; i <= unnn; ++i)
     {
      if ( unor[i] != k )
      continue;
      j = unixx[i];
      if ( ungsid[j] != ungsid[i] )
      continue;
      unqc[j] = unqc[j] + unqc[i];
     }
   }
  for (i = 0; i <= unnn; ++i)
   {
    unqct[i] = unqc[i];
   }
  /* Calculate sum of GS inflow  (inflows ?  */
  for (i = 0; i <= gsnn; ++i)
   {
    for (j = 0; j <= gsnn; ++j)
     {
      if ( gsnx[i] == gsid[j] )
      {
      gsqi[j] = gsqi[j] + gsqo[i];
      }
     }
   }
  /* 1) Estimate correction factors GSCF */
  /* 2) Assign correction factor for the UN watersheds which are
  /* outside GS zones, i.e., most downstream UN watersheds
  /* (next wsh for the last unit has GSID = 0) */

  for (i = 0; i <= gsnn; ++i)
   {
    x2 = rncf * gsvl[i] * xxx;
    gscf[i] = ( gsqo[i] / ( gsqi[i] + x2 ) - 1.0 ) /x2 ;
    if (gsnx[i] == 0 )
     {
      gscf[gsni] = gscf[i];
      gsid[gsni] = 0;
     }
   }

  /* Calculate cumulative flow (total= observed inflow + calculated
     cumulative flow */
  for (i = 0; i <= unnn; ++i )
   {
    k = i;
    j = unixx[k];
    l = ungsix[k];
```

```
    if ( ungsid[k] == ungsid[j] )  continue;
    if ( gsused[l] == 1 ) continue;
    xxx = gsqo[l];
    gsused[l] = 1;
    while (j < unni )
     {
      unqct[j] = unqct[j] + xxx;
      k = j;
      j = unixx[k];
      if ( ungsid[j] != ungsid[k] ) break;
     }
   }
  /* Final distribution of flow */
  for ( i = 0; i <= unnn; ++i)
   {
    j = ungsix[i];
    unqc[i] = unqct[i] * (1. + ( gscf[j] *unqc[i] ) );
   }

  /* Write results to output file */
  ftable2 = fopen (argv[3], "w");
  /* arc view version:
  fprintf ( ftable2, "\"%s\",\"%s\"\n", argv[4], argv[5]);
  */
  for (j = 0; j <= unnn; ++j )
    fprintf(ftable2, "%li,%f\n", unid[j], unqc[j] );
  fclose(ftable2);
  return 0;
 }
```

Appendix B Avenue scripts

This Appendix contains the following ArcView-Avenue scripts

1) Scripts that allow user to switch between different modules of the agrichemical transport model;

gotoapr	Opens a project that is specified in the first parameter
gtflwprc	Opens the project "flwprc.apr", (maps of the flow rate and precipitation depth), property: Click
gtflwu	Opens the project "flwprc.apr", property: Update
gtmodel	Opens the project “model.apr” (data preparation and model execution), property: Click
gtmodu	Opens the project “model.apr”, property: Update
gtresult	Opens the project “results.apr” (results of last model execution), property: Click
gtresu	Opens the project “results.apr”, property: Update
gttools	Opens the project “tools.apr” (selected tools for hydrologic maps preparation), property: Click
gttoou	Opens the project “tools.apr”, property: Update

2) Scripts from the project “model”;

edflow3	Displays the dialog box to select and modify the cumulative flow rate. Changes are made for the selected (or all) polygons active on the "Modeling Units" view. Assigned to the button EQc, property: Click.
edflow3u	Assigned to the button EQc (flow rate selection and modification), property: Update.

eduse2	Runs scripts "eduse2a" and "eduse2a" that support editing the agrichemical application rate. Assigned to button EAP. Property: Click.
eduse2a	Displays the dialog box to edit agrichemical application rate. Changes will be made to selected (or all) county polygons displayed on an active view (executed from the script "eduse2")
eduse2m	Displays the dialog box to edit agrichemical application rate. Changes will be made to selected (or all) modeling unit polygons displayed on active view (executed from the script "eduse2")
eduse2u	Assigned to button EAP (edit application rate). Property: update.
equat6	Calculates concentrations and loads for all or selected months of the year. Assigned to the button RUN, property: Click.
Selmodel	Displays the dialog box to select a model: equation: c = f(Flow, Area, Time, Use, X). Assigned to the button SMo, property: Click.
Stime1	Displays the dialog box to select a year (time variable for a model). Assigned to the button SYr, property: Click.
upwavq1	Calculates weighted average for upstream units. Incorporated into scripts eduse.
selup2	Selects upstream units (active theme must have the following fields: unit_id, unit_nx, and order). Assigned to a button in category: Tools, property: Apply.
Seldown2	Selects downstream units (active theme must have the following fields: unit_id, unit_nx, and order). Assigned to a button in category: Tools, property: Apply.
Selup2u	Update event for buttons that select upstream and downstream units ("selup2", "seldown2"). Assigned to a button in category: Tools, property: Update.
decay1	Calculates the concentrations and loads of a chemical that exponentially "decays" as it is carried by water. Assigned to the button FOR, property: Click.

3) Scripts from the project "model";

Cchart1	Draw charts of the concentration (12 months) at the center of selected features, property Click (runs script "schart1").
Lchart	Draw charts of the load (12 months) at the center of selected features, property Click (runs script "schart1").

Qchar1	Draw charts of the flow rate (12 months) at the center of selected features, property Click (runs script "schart1").

4) Scripts from the project “flwprc”;

pmchar1	Draws charts of the precipitation depth at the center of selected features, for selected months, property: Click, (executes script "schart1")
pmchar1u	Draws charts of the precipitation depth, property: Update
qmchar1	Draws charts of the flow rate at the center of selected features, for selected months, property: Click, (runs script "schart1").
qmchar1u	Draws charts of the flow rate, property: Update.
schart1	Draws charts, is executed from such scripts as pmchar1 and qmchar1 (Adopted from ESRII examples supplied with Avenue).
movie2	Displays charts in a sentence, category: tools, property: apply (script qmchar1u is used in the property: Update).
aliaset	Sets the alias names for fields, that describe a chart axis.

5) Scripts from the project “tools”;

order6	Determines the order of a stream/watershed in a flow system
upavg2	calculates weighted average over upstream units
cumul2	Accumulates values, going downstream (flow accumulation)
decom2	calculates the difference between the inflows and the outflow, the reverse process to the flow accumulation

B1 Changing ArcView projects from the PushButton Bar.

gotoapr Opens a project that is specified in the first parameter

```
'gotoapr.ave
gotoname = self.get(0)
theProject = av.GetProject
prname = theproject.getname
if (nil <> theProject) then
  if (theProject.IsModified) then
    res = MsgBox.SaveChanges("Do you want to save changes to " +
theProject.GetName + "?", "ArcView", true)
    if (nil = res) then exit end
    if (res) then
      av.Run("Project.Save", nil)
      if (theProject.IsModified) then exit end
    end
  end
end
xxx = theproject.getfilename.asstring.substitute(prname,gotoname)
theFName = xxx.asfilename
if (nil <> theProject) then
  theProject.Close
end
av.ClearGlobals
av.PurgeObjects
Project.Open(theFName)
```

gtflwprc Opens the project "flwprc.apr", (maps of the flow rate and precipitation depth), property: Click

```
xxx = av.run("gotoapr",{"flwprc.apr"})
```

gtflwu Opens the project "flwprc.apr", property: Update

```
prname = av.GetProject.getname
if(prname = "flwprc.apr") then
  self.setenabled(false)
else
  self.setenabled(true)
end
```

gtmodel **Opens the project “model.apr” (data preparation and model execution), property: Click**

```
xxx = av.run("gotoapr",{"model.apr"})
```

gtmodu **Opens the project “model.apr”, property: Update**

```
prname = av.GetProject.getname
if(prname = "model.apr") then
  self.setenabled(false)
else
  self.setenabled(true)
end
```

gtresult **Opens the project “results.apr” (results of last model execution), property: Click**

```
xxx = av.run("gotoapr",{"results.apr"})
```

gtresu **Opens the project “results.apr”, property: Update**

```
prname = av.GetProject.getname
if(prname = "results.apr") then
  self.setenabled(false)
else
  self.setenabled(true)
end
```

gttools **Opens the project “tools.apr” (selected tools for hydrologic maps preparation), property: Click**

```
xxx = av.run("gotoapr",{"tools.apr"})
```

gttoou **Opens the project “tools.apr”, property: Update**

```
prname = av.GetProject.getname
if(prname = "tools.apr") then
  self.setenabled(false)
else
  self.setenabled(true)
end
```

B2 Scripts from the project “model”

edflow3 Displays the dialog box to select and modify the cumulative flow rate. Changes are made for the selected (or all) polygons active on the "Modeling Units" view. Assigned to the button EQc, property: Click.

```
'edflow3.ave
'=====================================================
'data table name:   Cumulative flow
'         fields:   "Unit_id", m199001 ...
'model table name:  feature table of "units" ("Crwsd")
'          fields:  "Unit_id", "Qm01 .. Qm12"
'=====================================================
theView = av.getproject.finddoc("Modeling Units")
FTmodl = theview.findtheme("Units").Getftab
FTflow = av.getproject.finddoc("Cumulative flow").getVtab
'FTflow = fldoc.findtheme("unflow").getftab
if ((FTmodl = nil) or (FTflow=nil)) then
  msgbox.error("Can't get ftab of units or cumulative flow",
         " Units or flow unaccessible")
  exit
end
'check if all fields can be found:
'current version of edflow1 and v2 assumes that all fields exist
sp = "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
ln1 = "Assume flow conditions of:"
ln3 = "( to apply factor to current values enter: NO )"
ln4 = "Multiply flow rate for all months by:"
ln5 = "      Apply multiplication factor to each month:"
ln6 = "Multiply flow rate for January by......."
ln7 = "Multiply flow rate for February by....."
ln8 = "Multiply flow rate for March by.........."
ln9 = "Multiply flow rate for April by............"
ln10 = "Multiply flow rate for May by............"
ln11 = "Multiply flow rate for June  by.........."
ln12 = "Multiply flow rate for July  by............"
ln13 = "Multiply flow rate for August by........"
ln14 = "Multiply flow rate for September by."
ln15 = "Multiply flow rate for October by......"
ln16 = "Multiply flow rate for November by.."
ln17 = "Multiply flow rate for December by.."
labls = {ln1,ln3,ln4,sp,ln5,ln6,ln7,ln8,ln9,ln10,ln11,ln12,
         ln13,ln14,ln15,ln16,ln17}
defts = {"1990"," ","1.00"," ","
","1.00","1.00","1.00","1.00","1.00",
         "1.00","1.00","1.00","1.00","1.00","1.00","1.00"}
```

```
tytul = "    Select flow rate"
while (true)
  xlist = msgBox.Multiinput(tytul,
    "Monthly average flow rate m3/s", labls, defts )
  if (xlist.count = 0) then
    exit
  end
  for each i in 0..16
    if ((i=0) or (i=1) or (i=3) or (i=4))then
      continue
    end
    if (xlist.get(i).isnumber.not) then
      msgbox.error(xlist.get(i)++"is not a number",
      "Error in line "++i.asstring)
      continue
    end
  end
  if (xlist.get(0) = "no" )then
    break
  end
  yr = xlist.get(0).asnumber
  if ((yr < 1960) or (yr>1991)) then
    msgbox.error(xlist.get(0)++"out of range (1960-1991)",
      "Error in year field ")
      continue
  end
  break
end
if (xlist.get(0) = "no" ) then
  if(Ftmodl.iseditable.not) then
    Ftmodl.seteditable(true)
  end
  for each m in 1..12
    av.setstatus (m * 100 /12)
    av.showmsg("Processing month "++m.asstring++" ... ")
    if (m <10) then
      nameto = "Flow"+"0"+m.asstring
    else
      nameto = "Flow"+m.asstring
    end
    im = m + 4
    expr = "["+nameto+"] *"+xlist.get(2)+"*"+xlist.get(im)
    Ftmodl.calculate(expr,Ftmodl.findfield(nameto))
  end
  if(Ftmodl.iseditable) then
    Ftmodl.seteditable(false)
  end
  exit
end
' Find the table and join fields for the tables
field1 = Ftmodl.FindField("Unit_id")
field2 = Ftflow.FindField("Unit_id")
' Now perform the join....
Ftmodl.Join( field1, Ftflow, field2)
if(Ftmodl.iseditable.not) then
```

```
  Ftmodl.seteditable(true)
end
for each m in 1..12
  if (m <10) then
     namefrom = "Qm"+xlist.get(0)+"0"+m.asstring
     nameto = "Flow"+"0"+m.asstring
  else
     namefrom = "Qm"+xlist.get(0)+m.asstring
     nameto = "Flow"+m.asstring
  end
  im = m + 4
  expr = "["+namefrom+"] *"+xlist.get(2)+"*"+xlist.get(im)
  Ftmodl.calculate(expr,Ftmodl.findfield(nameto))
end
if(Ftmodl.iseditable) then
  Ftmodl.seteditable(false)
end
FTmodl.unjoinall
```

edflow3u **Assigned to the button EQc (flow rate selection and modification), property: Update.**

```
' edflow3u.ave : update event for "edit/select flow rate"
' Is enabled when  "Modeling Units" view is active.
' ======================
theView = av.GetActiveDoc
if (theView.GetEditableTheme <> NIL) then
  SELF.SetEnabled(FALSE)
  exit
end
t = theView.GetName
if (  t = "Modeling Units" ) then
   SELF.SetEnabled(TRUE)
   exit
end
SELF.SetEnabled(FALSE)
```

eduse2 **Runs scripts "eduse2a" and "eduse2a" that support editing the agrichemical application rate. Assigned to button EAP. Property: Click**.

```
'eduse2.ave
'================
t = av.GetActiveDoc.GetName
if ( t = "Modeling Units") then
  av.run("eduse2m","")
  exit
end
if (t = "Application Rate") then
```

```
  av.run("eduse2a","")
  exit
end
msgbox.Info("Modeling Units or Application Rate must be active",
            "Raquired View is not Active")
```

eduse2a Displays the dialog box to edit agrichemical application rate. Changes will be made to selected (or all) county polygons displayed on an active view (executed from the script "eduse2")

```
'eduse2a.ave (updated eduse1b.ave)
'edtuse.update script checked the existence of themes.
'====================================================
'link table name:   "County link unit file" extracted from
'                   feature table of "Useunit"
'         fields:   "Fips", "Unit_id", "area_km2"
'model table name:  feature table of Units (alias of "Crwsd")
'          fields:  "Unit_id", "Chemuse"
'county table name: feature table of Use feature from
'                   "Application Rate" view, (alias of "cruse")
'           fields: "Fips", "use"
'====================================================

theView = av.getproject.finddoc("Modeling Units")
FTmodl = theview.findtheme("Units").Getftab
FTlink = av.getproject.finddoc("linkuse.dbf").getvtab
FTcnty = av.getproject.finddoc("Application
Rate").findtheme("use").GetFtab
if ( Ftlink.haserror ) then
  msgbox.error ( " Can't find link table",
               "theme "++FTmodl.asstring)
  exit
end
'check if all fields can be found:
FDumdl = FTmodl.findfield("Unit_id")
if ( FDumdl = nil ) then
  msgbox.error ( " Can't find Unit_id field",
               "theme "++FTmodl.asstring)
  exit
end
FDfcty = FTcnty.findfield("Fips")
  if ( FDfcty = nil ) then
    msgbox.error ( " Can't find Fips field",
               "theme "++FTcnty.asstring)
    exit
  end
FDucty = FTcnty.findfield("use")
  if ( FDucty = nil ) then
    msgbox.error ( " Can't find Use field",
               "theme "++FTcnty.asstring)
    exit
  end
```

```
FDtcty = FTcnty.findfield("temp")
  if ( FDucty = nil ) then
    msgbox.error ( " Can't find Temp field",
               "theme "++FTcnty.asstring)
    exit
  end
FDclnk = FTlink.findfield("Fips")
  if ( FDclnk = nil ) then
    msgbox.error ( " Can't find Fips field",
               "theme "++FTlink.asstring)
    exit
  end
FDulnk = FTlink.findfield("Unit_id")
  if ( FDulnk = nil ) then
    msgbox.error ( " Can't find Unit_id field",
                 "theme "++FTlink.asstring)
    exit
  end
FDalnk = FTlink.findfield("area_km2")
  if ( FDalnk = nil ) then
    msgbox.error ( " Can't find Area_km2 field",
               "theme "++FTlink.asstring)
    exit
  end
'FDumdl = FTmodl.findfield("Unit_id")
FDrmdl = FTmodl.findfield("ChemUse")
  if ( FDrmdl = nil ) then
    msgbox.error ( " Can't find ChemUse field",
                 "theme "++FTmodl.asstring)
    exit
  end
'==========================================================
'This part is added for upstream weighetd average procedure
theFTab = FTmodl
FDunit = FDumdl
FDnext = theFtab.FindField("unit_nx")
if ( FDnext = nil ) then
   msgbox.error ( " Can't find unit_nx field",
             "theme "++FTmodl.asstring)
   exit
end
FDorder = theFtab.FindField("order")
if ( FDorder = nil ) then
   msgbox.error ( " Can't find order field",
             "theme "++FTmodl.asstring)
   exit
end
FDitem = FDrmdl
FDadd = theFtab.FindField("Cchemuse")
if ( FDadd = nil ) then
   msgbox.error ( " Can't find Cchemuse field",
             "theme "++FTmodl.asstring)
   exit
end
FDarea = theFtab.FindField("areakm2")
```

```
if ( FDarea = nil ) then
   msgbox.error ( " Can't find areakm2 field",
             "theme "++FTmodl.asstring)
   exit
end
parlst = {theFTab, FDunit, FDnext, FDorder, FDadd, FDitem, FDarea}
' av.run("upwavg1",parlst)
sp = "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
ln1 = "Take values from the database:"
ln2 = "(n89kgkm2, n90kgkm2,n91kgkm2,a89kgkm2)"
ln3 = "( to apply factor to current values enter: NO )"
ln4 = "Multiply data by factor:"
ln5 = "Apply specified value:"
ln6 = "( negative value deactivates this input field )"
ln7 = "( units:  nitrogen [kg/km2], atrazine [kg/km2])"
labls = {sp, ln1, ln2, ln3,sp ,ln4,sp,ln5,ln6,ln7}
multf = 1.000
userv = -1.0
defts = {" ","n89kgkm2"," "," "," ", multf.asstring," ",
userv.asstring," "," "}
tytul = "    Specify Application Rate"
while (true)
  xlist = msgBox.Multiinput(tytul,
    "Annual agrichemical application by county", labls, defts )
  if (xlist.count = 0) then
    exit
  end
  if (xlist.get(7).isnumber.not) then
    msgbox.error(xlist.get(7)++"is not a number", "Error in field
#3")
    continue
  end
  aaa = xlist.get(7).asnumber
  if (aaa >= 0) then
    'edit "application by county" (table in LTpar list) and exit
    'set cruse table editable
    if(FTcnty.iseditable.not) then
      FTcnty.seteditable(true)
    end
    FTcnty.calculate(xlist.get(7),FDucty)
    if(FTcnty.iseditable) then
      FTcnty.seteditable(false)
    end
    'exit
    break
  end
  'check multiplication factor
  if (xlist.get(5).isnumber.not) then
   msgbox.error(xlist.get(5)++"is not a number", "Error in field #2")
   continue
  end

  ' now check the historical record field (AV is not case sensitiwe !
  '  NO = no:
  if(xlist.get(1)="no") then
```

```
        'set cruse table editable
        if(FTcnty.iseditable.not) then
          FTcnty.seteditable(true)
        end
        expres = xlist.get(5)+"*[use]"
        FTcnty.calculate(expres,FDtcty)
        FTcnty.calculate("[temp]",FDucty)
        if(FTcnty.iseditable) then
          FTcnty.seteditable(false)
        end
        'exit
        break
      end
      'last possibility that user wants historical record !
      hrecord = FTcnty.findfield(xlist.get(1))
      if (hrecord = nil) then
        msgbox.error(xlist.get(1)++"  is neither NO nor field name"++ln2,
         "Error in field #1")
        continue
      end
      'set cruse table editable
      if(FTcnty.iseditable.not) then
        FTcnty.seteditable(true)
      end
      expres = xlist.get(5)+"*["+xlist.get(1)+"]"
      FTcnty.calculate(expres,FDucty)
      if(FTcnty.iseditable) then
        FTcnty.seteditable(false)
      end
      'exit
      break
    end  'of while error = false
    'make dictionaries:
    dlink = dictionary.make(FTlink.getnumrecords)
    'dc (dict.) relates fips and edited use (field "use")
    dc = dictionary.make(FTcnty.getnumrecords)
    'dm (dict.) relates unit_id and cumulative (use * area)
    dm = dictionary.make(FTmodl.getnumrecords)
    'da (dict.) relates unit id and cumulative area
    da = dictionary.make(FTmodl.getnumrecords)
    'assign initial values to cumulative (use * area) and (area)
    ' the average use over modelling unit = cum(use*area)/cum(area)
    for each rec in FTmodl
     dm.add(FTmodl.returnvalue(FDumdl,rec),0)
     da.add(FTmodl.returnvalue(FDumdl,rec),0)
    end
    'fill up the dc dictionary:  Fips ==> use (cruse counties coverage)
    for each rec in FTcnty
    dc.add(FTcnty.returnvalue(FDfcty,rec),FTcnty.returnvalue(FDucty,rec))
    end
    'calculate cumulative values
    for each rec in FTlink
      unitid = FTlink.returnvalue(FDulnk,rec)
      fips = FTlink.returnvalue(FDclnk,rec)
      area = FTlink.returnvalue(FDalnk,rec)
```

```
  applic = dc.get(fips)
  oldarea = da.get(unitid)
  oldappl = dm.get(unitid)
  da.set(unitid, oldarea + area )
  dm.set(unitid, oldappl + (area * applic))
end
'Write results into table "crwsd", item "Chemuse"
'set crwsd (model parameters) table editable
if(FTmodl.iseditable.not) then
  FTmodl.seteditable(true)
end
'Write results to "crwsd" table
for each rec in FTmodl
  unitid = FTmodl.returnvalue(FDumdl,rec)
  use = dm.get(unitid)/da.get(unitid)
  av.ShowMsg("Writing to chemuse: "++rec.asString++use.asstring)
  FTmodl.SetValue(FDrmdl, rec, use)
end
av.run("upwavg1",parlst)
'stop edit session (and refresh) "crwsd" table
if(FTmodl.iseditable) then
  FTmodl.seteditable(false)
end
```

eduse2m **Displays the dialog box to edit agrichemical application rate. Changes will be made to selected (or all) modeling unit polygons displayed on active view (executed from the script "eduse2")**

```
'eduse2m.ave (old eduse1c.ave)
'edtuse.update script checked the existence of themes.
'====================================================
'model table name:  feature table of "Units"
'          fields:  "Unit_id", "Chemuse"
'====================================================
theView = av.getproject.finddoc("Modeling Units")
FTmodl = theview.findtheme("Units").Getftab
if(theview.findtheme("Units").isactive.not) then
  msgbox.warning("Theme >Units< is not active", "")
end
'check if all fields can be found:
FDrmdl = FTmodl.findfield("ChemUse")
if ( FDrmdl = nil ) then
   msgbox.error ( " Can't find ChemUse field",
             "theme "++FTmodl.asstring)
   exit
end
'=========================================================
'This part is added for upstream weighetd average procedure
theFTab = FTmodl
FDunit = theFtab.FindField("unit_id")
if ( FDunit = nil ) then
   msgbox.error ( " Can't find unit_id field",
             "theme "++FTmodl.asstring)
   exit
end
FDnext = theFtab.FindField("unit_nx")
if ( FDnext = nil ) then
   msgbox.error ( " Can't find unit_nx field",
             "theme "++FTmodl.asstring)
   exit
end
FDorder = theFtab.FindField("order")
if ( FDorder = nil ) then
   msgbox.error ( " Can't find order field",
             "theme "++FTmodl.asstring)
   exit
end
FDitem = FDrmdl
FDadd = theFtab.FindField("Cchemuse")
if ( FDadd = nil ) then
   msgbox.error ( " Can't find Cchemuse field",
             "theme "++FTmodl.asstring)
   exit
end
FDarea = theFtab.FindField("areakm2")
if ( FDarea = nil ) then
   msgbox.error ( " Can't find areakm2 field",
             "theme "++FTmodl.asstring)
   exit
end
parlst = {theFTab, FDunit, FDnext, FDorder, FDadd, FDitem, FDarea}
' av.run("upwavg1",parlst)
'=======================================================
sp = "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
ln4 = "Multiply data by factor:"
ln5 = "Apply specified value:"
ln6 = "( negative value deactivates this input field )"
ln7 = "( units:  nitrogen [kg/km2], atrazine [kg/km2])"
labls = {sp, ln4,sp,ln5,ln6,ln7}
multf = 1.000
userv = -1.0
defts = {" ", multf.asstring," ", userv.asstring," "," "}
tytul = "    Specify Application Rate"
while (true)
  xlist = msgBox.Multiinput(tytul,
    "Annual agrichemical application by modeling unit",
      labls, defts )
  if (xlist.count = 0) then
    exit
  end
  if (xlist.get(3).isnumber.not) then
    msgbox.error(xlist.get(3)++"is not a number", "Error in field
#1")
    continue
  end
  aaa = xlist.get(3).asnumber
  if (aaa >= 0) then
```

```
    'put the user value into "ChemUse" field (selected or all recds)
    'set crwsd table editable
    if(FTmodl.iseditable.not) then
      FTmodl.seteditable(true)
    end
    FTmodl.calculate(xlist.get(3),FDrmdl)
    av.run("upwavg1",parlst)
    if(FTmodl.iseditable) then
      FTmodl.seteditable(false)
    end
    break
  end
  'check multiplication factor
  if (xlist.get(1).isnumber.not) then
    msgbox.error(xlist.get(1)++"is not a number", "Error in field
#1")
    continue
  end
  'set crwsd table editable
  if(FTmodl.iseditable.not) then
      FTmodl.seteditable(true)
  end
  expres = xlist.get(1)+"*[ChemUse]"
  FTmodl.calculate(expres,FDrmdl)
  av.run("upwavg1",parlst)
  if(FTmodl.iseditable) then
    FTmodl.seteditable(false)
  end
  break
end
```

eduse2u Assigned to button EAP (edit application rate). Property: update.

```
' eduse2u.ave : update event for "edit application"
' Is enabled when either "Application Rate" view or "Modeling Units"
' view is active.
' ======================
theView = av.GetActiveDoc
if (theView.GetEditableTheme <> NIL) then
  SELF.SetEnabled(FALSE)
  exit
end
t = theView.GetName
if ( ( t = "Modeling Units") or (t = "Application Rate") ) then
   SELF.SetEnabled(TRUE)
   exit
end
SELF.SetEnabled(FALSE)
```

equat6 **Calculates concentrations and loads for all or selected months of the year. Assigned to the button RUN, property: Click.**

```
'equat6 modified equat5, time component changed
'    into seasonal index, separate trend,
'    equation includes land-length and land-slope
'    model stored in model2.dbf
'equat4 calculates concentration and load in
'feature attibute table of "crwsd"
'model equations and parameters are stored in model1.dbf
'same as equat3 plus recalculation for selected months
'==================================================
theView = av.getproject.finddoc("Modeling Units")
FTmodl = theview.findtheme("Units").Getftab
if (FTmodl = nil) then
  msgbox.error("Can't find feature table", "Units unaccessible")
  exit
end
tab1 = av.getproject.finddoc("model2.dbf")
vtab1 = tab1.getvtab
if (vtab1 = nil) then
  msgbox.error("Can't find model2.dbf", "model2.dbf unaccessible")
  exit
end
i = -1
fsel = vtab1.findfield("sel")
for each rec in vtab1
  i = i + 1
  if (vtab1.returnvalue(fsel,rec) = 1 ) then
    idmod = i
    break
  end
end
fmodel = vtab1.findfield("Model")
xtxt = "Model:
"+vtab1.returnvalue(vtab1.findfield("Model"),idmod)++"?"
if(msgbox.miniYesNo(xtxt, True).not) then
      exit
end
'========= months selection ================================
yyy = "all OR selected months 0=NO, 1=YES"
mylist = {"1","0","0","0","0","0","0","0","0","0","0","0","0"}
lbs = {"All months . . . . . . . . . . . . . . . . . . . . . . . ",
    "January . . . . . . . . . . . . . . . . . . . . . . . . ",
    "February . . . . . . . . . . . . . . . . . . . . . . . ",
    "March  . . . . . . . . . . . . . . . . . . . . . . . . . ",
    "April   . . . . . . . . . . . . . . . . . . . . . . . . . ",
    "May  . . . . . . . . . . . . . . . . . . . . . . . . . . ",
    "June   . . . . . . . . . . . . . . . . . . . . . . . . . ",
    "July  . . . . . . . . . . . . . . . . . . . . . . . . .  ",
    "August . . . . . . . . . . . . . . . . . . . . . . . . ",
    "September  . . . . . . . . . . . . . . . . . . . .  ",
    "October . . . . . . . . . . . . . . . . . . . . . . .",
    "November . . . . . . . . . . . . . . . . . . . . .  ",
    "December . . . . . . . . . . . . . . . . . . . . . . . "}
xloop = true
while(xloop)
newlist = msgbox.multiinput(yyy, "Recalculate",lbs, mylist)
if(newlist.count = 0) then
  msgbox.info("Operation Canceled", "Process: RUN")
  exit
end
xloop = false
ind = 0
for each x in newlist
  ind = ind + 1
  if(x.isnumber.not) then
    msgbox.error("error in line"++ind.asstring+":"++x,"")
'    msgbox.info(x.asstring, "xxx")
    xloop = true
    mylist = newlist
    break
  end
end
end
if(newlist.get(0).asnumber > 0) then
  for each ind in 1..12
    newlist.set(ind, 1)
  end
else
  for each ind in 1..12
    newlist.set(ind, newlist.get(ind).asnumber)
  end
end
'========= end of months selection ========================
feq = vtab1.findfield("equation")
exp101 = vtab1.returnvalue(feq,idmod)
exp102 = exp101.substitute("U", "[Cchemuse]")
exp103 = exp102.substitute("A", "[Careakm2]")
exp104 = exp103.substitute("LS", "[Alndslp]")
exp105 = exp104.substitute("LL", "[Alndlgkm]")
' calculate trend coefficient
fyear = vtab1.findfield("year")
yr = vtab1.returnvalue(fyear,idmod)
xyr = yr.asstring
trfunc = vtab1.findfield("Ftrend")
trcoeff = vtab1.findfield("Trendcf")
exp400 = vtab1.returnvalue(trfunc,idmod)
exp401 = exp400.substitute("Year", xyr)
if(Ftmodl.iseditable.not) then
  Ftmodl.seteditable(true)
end
if(vtab1.iseditable.not) then
   vtab1.seteditable(true)
end
xxx0 = vtab1.calculate(exp401,trcoeff)
xtr = vtab1.returnvalue(trcoeff,idmod).asstring
exp110 = exp105.substitute("TR", xtr)
' clear fields:
```

```
'   xxx0 = vtab1.calculate("0",trcoeff)
av.showstopbutton
for each mt in 1..12
  xstop = av.setstatus (mt * 100 /12)
  av.showmsg("Processing month "++mt.asstring++" ... ")
  if(newlist.get(mt) = 0) then
    continue
  end
  if(xstop.not) then
     xyes =msgbox.Miniyesno("Do you want to stop ?", False)
     if(xyes) then
        if(Ftmodl.iseditable) then
           Ftmodl.seteditable(false)
        end
        if(vtab1.iseditable) then
           vtab1.seteditable(false)
        end
        av.clearstatus
        exit
     else
       av.ClearStatus
       av.showstopbutton
     end
  end
  xmt = mt.asstring
  if (mt <10) then
    Qname = "[Flow"+"0"+xmt+"]"
    Cname = "Conc"+"0"+xmt
    Lname = "Load"+"0"+xmt
    Sname = "Si"+"0"+xmt
  else
    Qname = "[Flow"+xmt+"]"
    Cname = "Conc"+xmt
    Lname = "Load"+xmt
    Sname = "Si"+xmt
  end
  'clear fields,
'  xxx0 = Ftmodl.calculate("0",Ftmodl.findfield(Cname))
'  xxx0 = Ftmodl.calculate("0",Ftmodl.findfield(Lname))
  xsi = vtab1.returnvalue(vtab1.findfield(Sname),idmod).asstring
  exp111 = exp110.substitute("SI", xsi)
  exp112 = exp111.substitute("Q", Qname)
  xxx3 = Ftmodl.calculate(exp112,Ftmodl.findfield(Cname))
  exp200 = "["+Cname+"] *"+Qname
  xxx4 = Ftmodl.calculate(exp200,Ftmodl.findfield(Lname))
end
if(Ftmodl.iseditable) then
  Ftmodl.seteditable(false)
end
if(vtab1.iseditable) then
   vtab1.seteditable(false)
end
av.clearstatus
av.clearmsg
```

Selmodel **Displays the dialog box to select a model: equation: c = f(Flow, Area, Time, Use, X). Assigned to the button SMo, property: Click.**

```
' selmodel (puts 1 into field "sel" of "model1.dbf"
' if selected, 0 otherwise.
'============================================================
tab1 = av.getproject.finddoc("model1.dbf")
vtab1 = tab1.getvtab
if (vtab1 = nil) then
  msgbox.error("Can't find model1.dbf", "model1.dbf unaccessible")
  exit
end
fsel = vtab1.findfield("sel")
fmodel = vtab1.findfield("Model")
modlist = list.make
for each rec in vtab1
  modlist.add(vtab1.returnvalue(fmodel,rec))
end
xselect = msgbox.choiceasstring(modlist,"Select model:",
    "Model selection")
if(xselect <> nil) then
  if(vtab1.iseditable.not) then
    vtab1.seteditable(true)
  end
  for each rec in vtab1
    if(xselect = vtab1.returnvalue(fmodel,rec)) then
       vtab1.setvaluenumber(fsel,rec,1)
    else
       vtab1.setvaluenumber(fsel,rec,0)
    end
  end
   if(vtab1.iseditable) then
     vtab1.seteditable(false)
  end
else
  msgbox.info("No selection made", "Nothing changed !")
end
```

Stime1 **Displays the dialog box to select a year (time variable for a model). Assigned to the button SYr, property: Click.**

```
'stime1
' writes year into the field "year" of "model1.dbf"
'=====================================================
tab1 = av.getproject.finddoc("model1.dbf")
vtab1 = tab1.getvtab
if (vtab1 = nil) then
  msgbox.error("Can't find model1.dbf", "model1.dbf unaccessible")
  exit
end
fyear = vtab1.findfield("Year")
if (fyear = nil) then
  msgbox.error("Can't find field: Year",
         "Field does not exist ?")
  exit
end
yrdef = vtab1.returnvalue(fyear,0).asstring
'prompt for year:
while(true)
  tx = "Specify a value for model's time variable (enter a year)"
  year = msgbox.input(tx,"Setting Model's Time Variable", yrdef)
  if(year = nil) then
    exit
  end
  if (year.isnumber.not) then
    msgbox.error(year++"is not a number",
      "Wrong entry ")
    continue
  end
  yr = year.asnumber
  if ((yr < 1900) or (yr>2999)) then
    msgbox.error(year++"out of range (1900-2999)",
      "Error in year field ")
      continue
  end
  break
end
if(vtab1.iseditable.not) then
  vtab1.seteditable(true)
end
vtab1.calculate(year,fyear)
if(vtab1.iseditable) then
  vtab1.seteditable(false)
end
```

upwavq1 Calculates weighted average for upstream units. Incorporated into scripts eduse.

```
' upwavg1.ave calculates weighted average for upstream units
' This is run from eduse... script
theFTab = self.get(0)
FDunit = self.get(1)
FDnext = self.get(2)
FDorder = self.get(3)
FDadd = self.get(4)
FDitem = self.get(5)
FDarea = self.get(6)
ltunit = list.make
ltnext = list.make
ltorder = list.make
```

```
ltitem = list.make
ltadd = list.make
unnx = theFTab.GetNumRecords
unnn = unnx - 1
dcarea = dictionary.make(unnx)
dcmass = dictionary.make(unnx)
for each rec in theFTab
  Nunit = theFTab.ReturnValueNumber(FDunit, rec)
  ltunit.Add(Nunit)
  ltnext.Add(theFTab.ReturnValueNumber(FDnext, rec))
  ltorder.Add(theFTab.ReturnValueNumber(FDorder, rec))
  Narea = theFTab.ReturnValueNumber(FDarea, rec)
  Nmass = theFTab.ReturnValueNumber(FDitem, rec) * Narea
  dcarea.add(Nunit, Narea )
  dcmass.add(Nunit, Nmass )
end
' find maximum order
unmaxord = 1
for each i in 0..unnn
  av.ShowMsg("Finding maximum order ..."++i.asString)
  istatus = i * 100 / unnn
  av.SetStatus(istatus)
  if (ltorder.get(i) > unmaxord ) then
        unmaxord = ltorder.get(i)
  end
end
'calculate cumulative values
unmaxord1 = unmaxord - 1
for each k in 1..unmaxord1
  av.ShowMsg("Calculating cumulative values ..."++k.asString)
  istatus = k * 100 / unmaxord
  av.SetStatus(istatus)
  for each i in 0..unnn
    if ( ltorder.get(i) <> k ) then
      continue
    end
    Nunit = ltunit.get(i)
    Nnext = ltnext.get(i)
    Nxmass = dcmass.get(Nunit) + dcmass.get(Nnext)
    Nxarea = dcarea.get(Nunit) + dcarea.get(Nnext)
    dcmass.set(Nnext, Nxmass)
    dcarea.set(Nnext, Nxarea)
  end
end
av.clearMsg
'write results to Ftab
for each rec in theFtab
  Nunit = theFTab.ReturnValueNumber(FDunit, rec)
  Xavg = dcmass.get(Nunit) / dcarea.get(Nunit)
  TheFtab.SetValue( FDadd, rec, Xavg )
end
theFtab.Refresh
if(TheFtab.isEditable) then
    theFtab.SetEditable(false)
end
```

selup2 **Selects upstream units (active theme must have the following fields: unit_id, unit_nx, and order). Assigned to a button in category: Tools, property: Apply.**

```
'three fields are required:
nm_id = "unit_id"
nm_nx = "unit_nx"
nm_or = "order"
theView = av.GetActiveDoc
theThemes = theView.GetActiveThemes
p = theView.GetDisplay.ReturnUserPoint
shiftk = System.IsShiftKeyDown
if (ShiftK) then
  op = #VTAB_SELTYPE_XOR
else
  op = #VTAB_SELTYPE_NEW
end
t = thethemes.get(0)
if (t.CanSelect.not) then
  exit
end
t.SelectByPoint(p, op)
recs = t.FindByPoint(p)
if(recs.count = 0) then
  exit
end
rec = recs.get(0)
v = t.getFTab
fid = v.findfield(nm_id)
fnx = v.findfield(nm_nx)
ford = v.findfield(nm_or)
if ((fid=nil)or(fnx=nil)or(ford=nil)) then
  msgbox.error ( " Can't find necessary fields",
                 " Upstream selection ")
  exit
end
xid = v.returnvalue(fid, rec)
xnx = v.returnvalue(fnx, rec)
rno = rec
rord = v.returnvalue(ford,rno)
if(rord = 1) then
  exit
end
bmap = v.getselection
stk = stack.make
av.showmsg("Selecting upstream elements ...")
while (true)
  for each rcd in v
    'don't look for upstream if first order
    if(rord = 1 ) then
      break
    end
    xnx = v.returnvalue(fnx,rcd)
    if(xid = xnx) then
      stk.push(rcd.asstring)
    end
  end
  rno = stk.pop.asnumber
  if(shiftk) then
    if(bmap.get(rno)) then
       bmap.clear(rno)
    else
       bmap.set(rno)
    end
  else
    bmap.set(rno)
  end
  t.blinkrecord(rno)
  rord = v.returnvalue(ford,rno)
  xid = v.returnvalue(fid,rno)
  if((stk.depth = 0 )and (rord = 1)) then
    break
  end
end
av.Clearmsg
```

Seldown2 **Selects downstream units (active theme must have the following fields: unit_id, unit_nx, and order). Assigned to a button in category: Tools, property: Apply.**

```
'seldown2
'three fields are required:
nm_id = "unit_id"
nm_nx = "unit_nx"
nm_or = "order"
theView = av.GetActiveDoc
theThemes = theView.GetActiveThemes
p = theView.GetDisplay.ReturnUserPoint
shiftk = System.IsShiftKeyDown
if (ShiftK) then
  op = #VTAB_SELTYPE_XOR
else
  op = #VTAB_SELTYPE_NEW
end
t = thethemes.get(0)
if (t.CanSelect.not) then
  exit
end
t.SelectByPoint(p, op)
recs = t.FindByPoint(p)
if(recs.count = 0) then
  exit
end
```

```
rec = recs.get(0)
v = t.getFTab
fid = v.findfield(nm_id)
fnx = v.findfield(nm_nx)
ford = v.findfield(nm_or)
if ((fid=nil)or(fnx=nil)or(ford=nil)) then
  msgbox.error ( " Can't find necessary fields",
               " Downstream selection ")
  exit
end
xid = v.returnvalue(fid, rec)
xnx = v.returnvalue(fnx, rec)
rno = rec
rord = v.returnvalue(ford,rno)
if(xnx = 0) then
  exit
end
bmap = v.getselection
av.showmsg("Selecting downstream elements ...")
while (true)
  for each rcd in v
    xid = v.returnvalue(fid,rcd)
    if(xid = xnx) then
      bmap.set(rcd)
      if(shiftk) then
        if(bmap.get(rcd)) then
          bmap.clear(rcd)
        else
          bmap.set(rcd)
        end
      else
        bmap.set(rcd)
      end
      xnx = v.returnvalue(fnx,rcd)
      if (xnx = 0) then
        exit
      end
    end
  end
end
msgbox.error("I cann't find the most downstream reach",
              "Last reach not found")
```

Selup2u **Update event for buttons that select upstream and downstream units ("selup2", "seldown2"). Assigned to a button in category: Tools, property: Update.**

```
'selup2.update
' Is enabled when prpoer fields are found in active
'three fields are required:
nm_id = "unit_id"
nm_nx = "unit_nx"
```

```
nm_or = "order"
theView = av.GetActiveDoc
if (theView.GetEditableTheme <> NIL) then
  SELF.SetEnabled(FALSE)
  exit
end
for each t in theView.GetThemes
  if (t.IsVisible.not) then
    continue
  end
  if (t.IsActive.not) then
    continue
  end
  fid = t.getftab.findfield(nm_id)
  fnx = t.getftab.findfield(nm_nx)
  ford = t.getftab.findfield(nm_or)
  if ((fid<>nil)and(fnx<>nil)and(ford<>nil)) then
     SELF.SetEnabled(TRUE)
     exit
  end
  break
end
SELF.SetEnabled(FALSE)
```

decay1 **Calculates the concentrations and loads of a chemical that exponentially "decays" as it is carried by water. Assigned to the button FOR, property: Click.**

```
' decay1.ave Calculates loads and concentrations
'            in surface waters. Chemical losses
'            are governed by the first order reaction.
'unit conversion factor
' Assumptions:
' application rate is in kg/km2/yr
' area is in km2
' then, the mass runoff is in kg/year
' to make load units g/s
' the conversion factor equal to 0.00003170979
' 1000 (g/kg) / ( 365 days * 86400 seconds )
' must be applied:
conversf = 0.00003170979
' flow is in m3/s, thus concentration is in g/m3
' or mg/L
' The usercoef represents seasonal index, confidence
' limits, extreme conditions, application timing etc.
' It is a multiplier of mass runoff from the field.
usercoef = 1

' This part is for an Attribute table !!!)
' note: dbf table contains a dummy (first) record
' that may influence calculations (unlikely)
' but will produce avenue errors (dividing by 0)
```

```
theView = av.getproject.finddoc("Modeling Units")
theFtab = theview.findtheme("Units").Getftab
TBunit = theFtab
' Ask user for needed fields:
TXunit = "unit_id"
TXnext = "unit_nx"
TXorder = "order"
TXarea = "Areakm2"
TXuse = "Chemuse"
TXtrat = "Travtime"
TXloss = "Losscoef"
TXexpt = "Expofac"
TXflow = "Flow"
FDunit = theFtab.FindField(TXunit)
FDnext = theFtab.FindField(TXnext)
FDorder = theFtab.FindField(TXorder)
FDarea = theFtab.FindField(TXarea)
FDuse = theFtab.FindField(TXuse)
FDtrat = theFtab.FindField(TXtrat)
FDloss = theFtab.FindField(TXloss)
FDexpt = theFtab.FindField(TXexpt)
FDflow = theFtab.FindField(TXflow)
' Make a list of VTunit fields
LSFDunit = theFtab.GetFields
' If the default fields can not be found, ask user for
' the required fields:
if(FDunit=nil) then
  ' Select current_unit id field (from entity)
  FDunit = MsgBox.List(LSFDunit, "Select a field that contains"
      ++NL++"the watershed/stream ID", "Unit ID")
end
' If not selected, quit
IF (FDunit = nil) then
  exit
end
if(FDnext=nil) then
  ' Select downstream_unit id field
  FDnext = MsgBox.List(LSFDunit, "Select a field that contains"
       ++NL++" the downstream unit ID", "Downstream unit ID")
end
' If not selected, quit
IF ( FDnext = nil) then
  exit
end
if(FDorder=nil) then
  ' Select order field
  FDorder = MsgBox.List(LSFDunit, "Select field that contains"
     ++NL++" the watershed/stream order","Order")
end
' If not selected, quit
IF ( FDorder = nil) then
  xxx = msgbox.info("Create a field of stream order",
       "Missing fields")
  exit
end
```

```
if(FDarea=nil) then
  ' Select field that contains area of the modeling unit
  FDarea = MsgBox.List(LSFDunit, "Select a field that contains"
  ++NL++"modeling unit area (km2)", "Area")
end
' If not selected, quit
IF ( FDarea = nil) then
  exit
end
if(FDuse=nil) then
  ' Select field that contains the application rate within unit
  FDuse = MsgBox.List(LSFDunit, "Select a field that contains"
  ++NL++"application rate (kg/km2/yr)", "Application Rate")
end
' If not selected, quit
IF ( FDuse = nil) then
  exit
end
if(FDexpt=nil) then
  ' Select field that contains the export factors
  FDexpt = MsgBox.List(LSFDunit, "Select a field that contains"
  ++NL++"the export factors", "Export factor")
end
' If not selected, quit
IF ( FDexpt = nil) then
  exit
end
if(FDtrat=nil) then
  ' Select field that contains travel time through modeling unit
  FDtrat = MsgBox.List(LSFDunit, "Select a field that contains"
  ++NL++"unit travel time (day)", "Travel Time")
end
' If not selected, quit
IF ( FDtrat = nil) then
  exit
end
if(FDloss=nil) then
  ' Select field that contains the loss coefficient
  FDloss = MsgBox.List(LSFDunit, "Select a field that contains"
  ++NL++"the loss coefficient (1/d)", "Loss coefficient")
end
' If not selected, quit
IF ( FDloss = nil) then
  exit
end
if(FDflow=nil) then
  ' Select field that contains the accumulated flow rate
  FDflow = MsgBox.List(LSFDunit, "Select a field that contains"
  ++NL++"the total flow rate", "Flow rate")
end
' If not selected, quit
IF ( FDflow = nil) then
  exit
end
' Prompt for a name (name of the field, that will be created)
```

```
TXconc = msgbox.input("Enter the name of concentration field ",
"Concentration", "conc_gm3")
FDconc = theFtab.FindField(TXconc)
xx = false
if (FDconc = nil) then
  xx = msgbox.YesNo( TXconc++"field is not in the table "
  ++TBunit.asString+nl+"Do you want to create this field ?",
   "Field not Found !", true)
  if (xx.not) then
    exit
  end
  fenum = FDflow.gettype
  fprec = 6
  fwidth = 12
  xxx = msgbox.input("Please specify a width of the conc. field",
   "Width of the new field" , fwidth.asstring)
  fwidth = xxx.asnumber
  IF ( fwidth = nil) then
    exit
  end
  xxx2 = msgbox.input("Please specify number of decimal places:",
   "Precision of the new field" , fprec.asstring)
  fprec = xxx2.asnumber
  IF ( fprec = nil) then
    exit
  end
  'check if the theFtab can be edited if not, set it editable
  if(theFtab.isEditable.not) then
    theFtab.SetEditable(true)
  end
  ' Create a new field
  if(theFtab.CanAddFields) then
    FDconc = field.Make(TXconc,fenum,fwidth,fprec)
    theFtab.AddFields({FDconc})
  else
   msgBox.warning("Can not add a field to"++theFtab.asString,
                   "Can not edit table")
    exit
  end
else
  if(theFtab.isEditable.not) then
    theFtab.SetEditable(true)
  end
end
' Prompt for a name (name of the field, that will be created)
TXload = msgbox.input("Enter the name of load field ",
"Load", "load_gm3")
FDload = theFtab.FindField(TXload)
xx = false
if (FDload = nil) then
  xx = msgbox.YesNo( TXload++"field is not in the table "
  ++TBunit.asString+nl+"Do you want to create this field ?",
   "Field not Found !", true)
  if (xx.not) then
    exit
  end
  fenum = FDflow.gettype
  fprec1 = 6
  fwidth1 = 16
  xxx3 = msgbox.input("Please specify a width of the load field",
   "Width of the new field" , fwidth1.asstring)
    fwidth1 = xxx3.asnumber
  IF ( fwidth1 = nil) then
    exit
  end
  xxx4 = msgbox.input("Please specify number of decimal places:",
   "Precision of the new field" , fprec1.asstring)
    fprec1 = xxx4.asnumber
  IF ( fprec1 = nil) then
    exit
  end
  'check if the theFtab can be edited if not, set it editable
  if(theFtab.isEditable.not) then
    theFtab.SetEditable(true)
  end
  ' Create a new field
  if(theFtab.CanAddFields) then
    FDload = field.Make(TXload,fenum,fwidth1,fprec1)
    theFtab.AddFields({FDload})
  else
   msgBox.warning("Can not add a field to"++theFtab.asString,
                   "Can not edit table")
    exit
  end
else
  if(theFtab.isEditable.not) then
    theFtab.SetEditable(true)
  end
end
'the conversion factor and a multiplier
while(true)
  TXcf = msgbox.input("Enter a conversion factor"++NL++
            "(kg/yr -> g/s)", "Conversion Factor",
            conversf.asstring)
  if(TXcf = nil) then
    exit
  end
  if(TXcf.isnumber.not) then
    msgbox.error(TXcf++"is not a number","Wrong entry")
    continue
  end
  conversf = TXcf.asnumber
break
end
while(true)
  TXum = msgbox.input("Enter a multiplier of the"++NL++
            "chemical runoff from the field",
            "Runoff Multiplier", usercoef.asstring)
  if(TXum = nil) then
    exit
  end
  if(TXum.isnumber.not) then
    msgbox.error(TXum++"is not a number","Wrong entry")
    continue
  end
  usercoef = TXum.asnumber
break
end
'create lists
ltunit = list.make
ltnext = list.make
ltorder = list.make
'ltadd = list.make
'ltexnkt = list.make
'ltroff = list.make
unnx = theFTab.GetNumRecords
unnn = unnx - 1
dcload = dictionary.make(unnx)
dcexnkt = dictionary.make(unnx)
for each rec in theFTab
  Nunit = theFTab.ReturnValueNumber(FDunit, rec)
  ltunit.Add(Nunit)
  ltnext.Add(theFTab.ReturnValueNumber(FDnext, rec))
  ltorder.Add(theFTab.ReturnValueNumber(FDorder, rec))
  xk = theFTab.ReturnValueNumber(FDloss, rec)
  xt = theFTab.ReturnValueNumber(FDtrat, rec)
  exnkt = number.geteuler^( -1 * xk * xt )
  dcexnkt.add(Nunit, exnkt )
  xa = theFTab.ReturnValueNumber(FDarea, rec)
  xu = theFTab.ReturnValueNumber(FDuse, rec)
  xe = theFTab.ReturnValueNumber(FDexpt, rec)
  xroff = xa * xu * xe * usercoef * conversf
  dcload.add(Nunit, xroff )
end
' find maximum order
unmaxord = 1
for each i in 0..unnn
  av.ShowMsg("Finding maximum order ..."++i.asString)
  istatus = i * 100 / unnn
  av.SetStatus(istatus)
  if (ltorder.get(i) > unmaxord ) then
        unmaxord = ltorder.get(i)
  end
end
'calculate values of load
unmaxord1 = unmaxord - 1
for each k in 1..unmaxord1
  av.ShowMsg("Decaying and decaying ..."++k.asString)
  istatus = k * 100 / unmaxord
  av.SetStatus(istatus)
  for each i in 0..unnn
    if ( ltorder.get(i) <> k ) then
      continue
    end
    Nunit = ltunit.get(i)
    Nnext = ltnext.get(i)
Nxvalue = dcload.get(Nunit) * dcexnkt.get(Nnext) + dcload.get(Nnext)
    dcload.set(Nnext, Nxvalue)
  end
end
av.clearMsg
av.clearStatus
'write results to Ftab
for each rec in theFtab
  Nunit = theFTab.ReturnValueNumber(FDunit, rec)
  Nload = dcload.get(Nunit)
  TheFtab.SetValue( FDload, rec, Nload )
  Nflow = theFTab.ReturnValueNumber(FDflow, rec)
  if (Nflow = 0 ) then
    Nconc = 0
  else
    Nconc = Nload / Nflow
  end
  TheFtab.SetValue( FDconc, rec, Nconc )
end
theFtab.Refresh
if(TheFtab.isEditable) then
    theFtab.SetEditable(false)
end
```

B3 Scripts from the project “results”

Cchart1 Draw charts of the concentration (12 months) at the center of selected features, property Click (runs script "schart1")

```
' Cchart1, One year of Concentrations from crwsd
chtitle = msgbox.input("Enter a legend's title",
        "Concentration Charts", "Conc. (m)g/m3")
if (chtitle <> nil ) then
  LTpar = {chtitle,"Conc",true,1978,1979, 1, 12}
  av.run("schart1",LTpar)
end
```

Lchart Draw charts of the load (12 months) at the center of selected features, property Click (runs script "schart1").

```
' Lchart1, One year of load from crwsd
chtitle = msgbox.input("Enter a legend's title",
        "Load Charts", "Load (m)g/s")
if (chtitle <> nil ) then
  LTpar = {chtitle, "Load", true, 1978, 1979, 1, 12}
  av.run("schart1",LTpar)
end
```

Qchar1 Draw charts of the flow rate (12 months) at the center of selected features, property Click (runs script "schart1").

```
' Qchart1, One year of flow from crwsd
chtitle = msgbox.input("Enter a legend's title",
        "Flow Charts", "Flow m3/s")
if (chtitle <> nil ) then
  LTpar = {chtitle, "Flow", true, 1978, 1979, 1, 12}
  av.run("schart1",LTpar)
end
```

B4 Scripts from the project “flwprc”

pmchar1 Draws charts of the precipitation depth at the center of selected features, for selected months, property: Click, (executes script "schart1")

```
' PMchar1,  displays multiple years of precipitation
'                  depth from gsprec or unprec
'=====================================================
'' title, prefix, one_year, fy, ty, fm, tm,
'=====================================================
sp = "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
ln1 = "Legend title:"
ln3 = "From year (min 1960):"
ln4 = "From month:"
ln5 = "To year (max 1991)"
ln6 = "To month"
labls = {ln1,ln3,ln4,ln5,ln6}
defts = {"Precip. cm/d", "1990","1","1990","12"}
tytul = "  Specify title and time interval"
while (true)
  xlist = msgBox.Multiinput(tytul,
    "Draw charts of monthly average precipitation", labls, defts )
  if (xlist.count = 0) then
    exit
  end
  for each i in 0..4
    if (i=0)then
      continue
    end
    if (xlist.get(i).isnumber.not) then
      msgbox.error(xlist.get(i)++"is not a number",
      "Error in line "++i.asstring)
      continue
    end
  end
  fy = xlist.get(1).asnumber
  ty = xlist.get(3).asnumber
  fm = xlist.get(2).asnumber
  tm = xlist.get(4).asnumber
  if (fy < 1960) then
    msgbox.error(xlist.get(1)++" - year out of range ( < 1960 )",
      "Error in year field ")
      continue
  end
  if (ty > 1991) then
    msgbox.error(xlist.get(3)++" - year out of range ( 1991 < )",
      "Error in year field ")
      continue
  end
  if ((fm < 1) or (fm>12)) then
    msgbox.error(xlist.get(3)++" - month out of range (1-12)",
      "Error in year field ")
      continue
  end
   if ((tm < 1) or (tm>12)) then
    msgbox.error(xlist.get(4)++" - month out of range (1-12)",
      "Error in year field ")
      continue
  end
  break
end
  LTpar = {xlist.get(0), "PM", false, fy, ty, fm, tm}
  av.run("schart1",LTpar)
```

pmchar1u Draws charts of the precipitation depth, property: Update

```
'Pmchar1u.ave
'multiple years data
'update event for "draw charts of precipitation"
' Is enabled when a theme is active and first
theView = av.GetActiveDoc
if (theView.GetEditableTheme <> NIL) then
  SELF.SetEnabled(FALSE)
  exit
end
xx = theView.GetActiveThemes
if(xx.count <> 0) then
  visnm = xx.get(0).getname
  if (( visnm = "unprec") or (visnm = "gsprec")) then
     SELF.SetEnabled(TRUE)
     exit
  end
end
SELF.SetEnabled(FALSE)
```

qmchar1 Draws charts of the flow rate at the center of selected features, for selected months, property: Click, (runs script "schart1",)

```
' QMchar1,  multiple years of flow from gsflow or unflow
'=====================================================
'' title, prefix, one_year, fy, ty, fm, tm,
'=====================================================
sp = "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
ln1 = "Legend title:"
ln3 = "From year (min 1960):"
ln4 = "From month:"
ln5 = "To year (max 1991)"
```

```
ln6 = "To month"
labls = {ln1,ln3,ln4,ln5,ln6}
defts = {"Flow m3/s", "1990","1","1990","12"}
tytul = "  Specify title and time interval"
while (true)
  xlist = msgBox.Multiinput(tytul,
    "Draw charts of monthly average flow rate", labls, defts )
  if (xlist.count = 0) then
    exit
  end
  for each i in 0..4
    if (i=0)then
      continue
    end
    if (xlist.get(i).isnumber.not) then
      msgbox.error(xlist.get(i)++"is not a number",
      "Error in line "++i.asstring)
      continue
    end
  end
  fy = xlist.get(1).asnumber
  ty = xlist.get(3).asnumber
  fm = xlist.get(2).asnumber
  tm = xlist.get(4).asnumber
  if (fy < 1960) then
    msgbox.error(xlist.get(1)++" - year out of range ( < 1960 )",
      "Error in year field ")
      continue
  end
  if (ty > 1991) then
    msgbox.error(xlist.get(3)++" - year out of range ( 1991 < )",
      "Error in year field ")
      continue
  end
  if ((fm < 1) or (fm>12)) then
    msgbox.error(xlist.get(3)++" - month out of range (1-12)",
      "Error in year field ")
      continue
  end
   if ((tm < 1) or (tm>12)) then
    msgbox.error(xlist.get(4)++" - month out of range (1-12)",
      "Error in year field ")
      continue
  end
  break
end
theView = av.GetActiveDoc
lstact = theView.GetActiveThemes
for each t in lstact
  if ( t.getname = "unflow") then
    LTpar = {xlist.get(0), "Qm", false, fy, ty, fm, tm}
    av.run("schart1",LTpar)
    exit
  end
  if ( t.getname = "gsflow") then
```

```
    LTpar = {xlist.get(0), "M", false, fy, ty, fm, tm}
    av.run("schart1",LTpar)
    exit
  end
end
```

qmchar1u Draws charts of the flow rate, property: Update

```
'Qmchar1u.ave
'multiple years data
'update event for "draw charts of flow"
' Is enabled when the theme is active and first
theView = av.GetActiveDoc
if (theView.GetEditableTheme <> NIL) then
  SELF.SetEnabled(FALSE)
  exit
end
xx = theView.GetActiveThemes
if(xx.count <> 0) then
  visnm = xx.get(0).getname
  if (( visnm = "unflow") or (visnm = "gsflow")) then
     SELF.SetEnabled(TRUE)
     exit
  end
end
SELF.SetEnabled(FALSE)
```

schart1 Draws charts, is executed from such scripts as pmchar1 and qmchar1 (Adopted from ESRII examples supplied with Avenue)

```
' MakeBarChartSpotSymbols
' title, prefix, one_year, fy, ty, fm, tm,
ltitle = self.get(0)
prefix = self.get(1)
one_year = self.get(2)
fy = self.get(3)
ty = self.get(4)
fm = self.get(5)
tm = self.get(6)
' title for legend
'ltitle = "Flow m3/s"
' prefix (field name = prefix + 01..12  or prefix + yr + mt)
'prefix = "M"
' one year will be displayed ???
'one_year = false
'fy = 1988
'fm = 1
'ty = 1990
'tm = 12
' optional legend color
if(one_year) then
```

```
  fy = 1989
  ty = 1989
  fm = 1
  tm = 12
end
'Script.The.SetNumberFormat("d.ddddd")
'The max. bar height and width are calculated as perc. of height of
'displayed area. The largest bar is drawn at this height and the rest
'are scaled accordingly. A value of 0.1 means the largest bar will be
'0.1 of the height of the current display. Four sizes (small, medium,
'large and x-large) are programmed as examples.
bar_size = MsgBox.ListAsString ({"Small","Medium","Large","X-
Large"},"Select bar size:","Bar Symbols")
if (bar_size = "Small") then
  max_bar_height = 0.1
  bar_width = 0.01
  legend_gridlines = 2
elseif (bar_size = "Medium") then
  max_bar_height = 0.2
  bar_width = 0.02
  legend_gridlines = 3
elseif (bar_size = "Large") then
  max_bar_height = 0.3
  bar_width = 0.03
  legend_gridlines = 4
elseif (bar_size = "X-Large") then
  max_bar_height = 0.4
  bar_width = 0.04
  legend_gridlines = 5
else
  exit  'Cancel selected'
end
'Number of gridlines on legend bar. Uncomment the following line
'to override above settings.
legend_gridlines = 4
legend_color = Color.GetCyan
'draw_gridlines = MsgBox.YesNo("Draw gridlines on the bars?","Bar
Symbols",False)
draw_gridlines = true
'Point size of text used in legend
leg_text_size = 8
'To ignore specific data values (eg., those that mean "nodata") put
'the values in this list and they will not be charted. This list
'should always contain 0, to prevent drawing zero size bars.
ignore_values = {0}
'List of colors
'Make your own custom colors here and make sure to add to both
'the color_list and also color_names...
pink = Color.Make
pink.SetRGBList({255,105,180})
olive = Color.Make
olive.SetRGBList({188,238,104})
orange = Color.Make
orange.SetRGBList({255,69,0})
gold = Color.Make
```

```
gold.SetRGBList({255,215,0})
maroon = Color.Make
maroon.SetRGBList({255,52,179})
color_list = {Color.GetBlue, Color.GetCyan, gold,
              Color.GetGray, Color.GetGreen, Color.GetMagenta,
maroon,
              olive, orange, pink, Color.GetRed, Color.GetYellow }
color_names = {"Blue", "Cyan", "Gold", "Gray", "Green", "Magenta",
               "Maroon", "Olive", "Orange", "Pink",
               "Red", "Yellow"}
'Number of colors
num_colors = 12
theView = av.GetActiveDoc
theProjection = theView.GetProjection
project_flag = theProjection.IsNil.Not  'true if projected
theTheme = theView.GetActiveThemes.Get(0)  'Get first active theme
theFTab = theTheme.GetFTab
shpfield = theFtab.FindField("Shape")
gra_list = theView.GetGraphics
'This section prompts the user for the fields in the
'FTab to generate the bar symbols from.
f= " "
max_field_prec = 0      'Field precision of selected fields
selected_fields = {}    'Fields selected by user
numeric_fields = {}     'Numeric fields in the FTab
field_aliases = {}      'List of field aliases
all_fields = theFtab.GetFields
'Build list of numeric fields from all fields.
'Also list of field aliases to display to user.
for each f in all_fields
  if (f.IsTypeNumber) then
    numeric_fields.Add(f)
    field_aliases.Add(f.GetAlias)
  end
end  'for
'Select fields for bar symbols
user_colors = {}
done = FALSE
field_count = 0
for each yr in fy..ty
  for each mt in 1..12
    if (( yr = fy) and ( mt < fm)) then
      continue
    end
    if (( yr = ty) and ( mt > tm)) then
      continue
    end

    if (mt < 10 ) then
      txt = "0"+mt.AsString
    else
      txt = mt.AsString
    end
    if(one_year) then
      fname = prefix+txt
    else
      fname = prefix+yr.AsString+txt
    end
  if (fname <> nil) then
    field_count = field_count + 1
    f = theFTab.findfield(fname)
    selected_fields.Add(f)
    user_colors.Add(color_list.Get(field_count - 1 mod num_colors))
    'Store max field precision of selected fields for later use.
    max_field_prec = (max_field_prec Max f.GetPrecision)
  else
    'Cancel selected.
    done = TRUE
  end 'if
end   'month
end   'year
field_count = selected_fields.Count
if (field_count = 0) then
  'User did not select any fields to symbolize
  exit
end
'Set bar height and width in page display units
view_extent = theView.GetDisplay.ReturnVisExtent
max_bar_height = view_extent.GetHeight * max_bar_height
bar_width = view_extent.GetHeight * bar_width
'Iterate thru selection bitmap or all records in FTab if no
selection.
iter = theFtab.GetSelection
num_records = iter.Count
'If no selected set, set iteration to all records in FTab.
if (num_records = 0) then
  iter = theFTab
  num_records = theFTab.GetNumRecords
end
min_height = 0
max_height = 0
for each rec in iter
  for each f in selected_fields
    cur_height = theFTab.ReturnValueNumber(f,rec)
    if (ignore_values.FindByValue(cur_height) > -1) then
      'Ignore values user wants to ingore
      continue
    end
    min_height = (min_height Min cur_height)
    max_height = (max_height Max cur_height)
  end  'for
end   'for
'Since the max height of a bar could be negative, find abs value of
'both min and max values to determine a scale factor. When building
the
'actual bar for a field, divide field value by this scale factor.
That
'way the maximum bar height (max_bar_height) will = the max field
value,
'whether positive or negative; all other field values are scaled
accordingly.
ht_scale_factor = (min_height.Abs) Max (max_height.Abs)
'Figure out break points for legend to allow adding gridlines to
legend
'and (optionally) actual bars on map. This set of code figures out a
'round number larger than the maximum field value, then divides that
'number by the number of legend classes desired. Then it rounds the
break
'point to a "nice" round number. E.g., given a max field value of
123,
'and 3 classes, this set of code will create a legend with break
points
'at 45, and a top value of 135.
if (ht_scale_factor < 1) then
  theLog = ht_scale_factor.Log(10).Floor
else
  theLog = ht_scale_factor.Log(10).Truncate
end  'if
x = ht_scale_factor / (10^(theLog))
y = (10 * x + 0.5).round * 10.^(theLog) / 10
x = y / legend_gridlines
if (x < 1) then
  theLog = x.Log(10).Floor
else
  theLog = x.Log(10).Truncate
end  'if
x2 = x / (10^(theLog))
legend_break_val = (2 * x2 + 0.5).round * 10.^(theLog) / 2
'Map value to display page units
legend_break_height = legend_break_val / ht_scale_factor *
max_bar_height
'Start drawing bar symbols
gra_list.UnselectAll
av.ShowStopButton
av.ShowMsg("Creating bar symbols...")
counter = 0
gra_group = GraphicGroup.Make
'For each selected record make the bars
for each rec in iter
  acolor = -1  'initialize color index
  'find the center of the feature to place bar symbol at
  ctr = theFtab.ReturnValue(shpfield,rec).ReturnCenter
  if (project_flag) then
    'Project the centroid if the view has a projection
    ctr = ctr.ReturnProjected(theProjection)
  end
  xcen = ctr.GetX
  ycen = ctr.GetY
  'Add line signaling zero in bar chart
  line_start_pt = (xcen - ((field_count * bar_width) / 2))@ycen
  line_end_pt = line_start_pt + ((bar_width * field_count)@0)
  aline = Line.Make(line_start_pt,line_end_pt)
  gr = GraphicShape.Make(aline)
  gra_group.Add(gr)
  'set starting point for drawing bars
  bar_start_pt = line_start_pt
  bar_count = 0  'number of bars drawn
  for each f in selected_fields
    bar_count = bar_count + 1
    acolor = acolor + 1
    'Convert field value to bar height;divide by height scale factor.
    thevalue = theFTab.ReturnValueNumber(f,rec)
    aheight = (thevalue / ht_scale_factor) * max_bar_height
    if (ignore_values.FindByValue(thevalue) > -1) then
      'Don't draw values user wants to ingore
      bar_start_pt = bar_start_pt + (bar_width@0)
      continue
    end  'if
    'draw the bar for the field
    abox = rect.Make(bar_start_pt,bar_width@aheight)
    gr = GraphicShape.Make(abox)
    'the symbol of a poly is a rasterfill, thus SetStyle request
works.
    gr.GetSymbol.SetStyle(#RASTERFILL_STYLE_SOLID)
    gr.GetSymbol.SetColor(user_colors.Get(acolor))
    gra_group.Add(gr)
    'Draw reference lines on bar to indicate value
    if (draw_gridlines) then
      num_tics = (aheight.Abs / legend_break_height).Floor
      c = 0
      while (c <> num_tics)
        c = c + 1
        if (aheight < 0) then
          tic_start_pt = bar_start_pt - (0@(c * legend_break_height))
        else
          tic_start_pt = bar_start_pt + (0@(c * legend_break_height))
        end  'if
        tic_end_pt = tic_start_pt + (bar_width@0)
        aline = Line.Make (tic_start_pt,tic_end_pt)
        gr = GraphicShape.Make(aline)
        gra_group.Add(gr)
      end  'while
    end  'if
    bar_start_pt = bar_start_pt + (bar_width@0)
  end  'for each f in selected_fields
  'Clone bar chart and add it to the graphic list. Then reuse graphic
group.
  'AddBatch used so that once ALL bar symbols for ALL records are
created
  'they are drawn to the screen. Using Add request is much slower.
  gg = gra_group.Clone
  gra_list.AddBatch(gg)
  'Add bar chart to the Theme's graphic list so that when the
  'theme is not drawn, the bar symbols won't draw either.
  theTheme.GetGraphics.Add(gg)
  gra_group.Empty
  counter = counter + 1
  progress = (counter / num_records) * 100
  more = av.SetStatus(progress)
  if (Not more) then
    'Break if user presses Stop button.
    break
  end
end  ' for each rec in iter
gra_list.UnselectAll
if (not ((num_records = 1) and (aheight = 0))) then
  'Special case where only 1 record selected, and the total of the
fields
  'of the record is zero; essentially no bars are drawn, so endbatch
  'should not be executed.
  gra_list.EndBatch
end
'Start building the legend.
'Set text size for legend
text_sym = av.GetSymbolWin.ReturnCurrentSymbol(#SYMBOL_TEXT).clone
text_sym.SetSize (leg_text_size)
'Add dummy text to get an idea for spacing of graphic elements
'Spacing of objects in legend = current point size of text.
gr = GraphicText.Make("ABC",0@0)
gr.SetSymbol (text_sym)
gr.SetSelected(TRUE)
gra_list.Add(gr)
spacing = gra_list.ReturnSelectedExtent.GetHeight
gra_list.RemoveGraphic(gr)
'Set number of decimal places for legend text equal to max field
precision
'This code builds a string like "d.ddd"
numformat = "d."
if (max_field_prec <> 0) then
  for each i in 1..max_field_prec
    numformat = numformat+"d"
  end
end  'if
'Draw legend at (0,0) then move it to the correct location
pnt = 0@0
abox = Rect.Make (pnt,bar_width@(legend_gridlines *
legend_break_height))
gr = GraphicShape.Make(abox)
gr.GetSymbol.SetStyle(#RASTERFILL_STYLE_SOLID)
gr.GetSymbol.SetColor(legend_color)
gra_group.Add(gr)
'Add "0" text in legend
gr = GraphicText.Make(0.SetFormat(numformat).AsString,
  (pnt + ((bar_width + spacing)@(-spacing / 2))))
gr.SetSymbol (text_sym)
gra_group.Add(gr)
'Draw a reference lines on bar.
for each i in 1..legend_gridlines
  line_start_pt = 0@(i * legend_break_height)
  line_end_pt = line_start_pt + (bar_width@0)
  aline = Line.Make (line_start_pt,line_end_pt)
  gr = GraphicShape.Make(aline)
  gra_group.Add(gr)
  'Add text indicating value represented by square
  atext = (legend_break_val * i).SetFormat(numformat).AsString
  'Add text string to right of line
  gr = GraphicText.Make(atext,(line_end_pt + (spacing@(-spacing /
2))))
  gr.SetSymbol (text_sym)
  gra_group.Add(gr)
  'gra_group.SetSelected (TRUE)
  'Clone graphic group;otherwise it would get deleted on Empty
  'gra_list.Add(gra_group.clone)
  'gra_group.Empty
end  'for
gra_group.SetSelected (TRUE)
'Clone graphic group;otherwise it would get deleted on Empty
gra_list.Add(gra_group.clone)
gra_group.Empty
gra_list.UngroupSelected   'Workaround
gra_list.GroupSelected
extent = gra_list.ReturnSelectedExtent
x = extent.ReturnOrigin.GetX
y = extent.GetTop + spacing
gr = GraphicText.Make(ltitle,(0@y))
'gr = GraphicText.Make("Flow Rate",(x@y + (x@0)))
'gr = GraphicText.Make("TEXT EXAMPLE",(x@y + ((spacing * 2)@0)))
gr.SetSymbol (text_sym)
gra_group.Add(gr)
'  acolor = acolor + 1   'Get next color
'  y = y + (spacing * 2) 'Move y up for next box
'end  'for
gra_group.SetSelected(TRUE)
gra_list.Add(gra_group)
gra_list.UngroupSelected  'Workaround
gra_list.GroupSelected
'Draw white box to place legend in
extent = gra_list.ReturnSelectedExtent
origin = extent.ReturnOrigin - (spacing@spacing)
size = extent.ReturnSize + ((2*spacing)@(2*spacing))
arect = Rect.Make(origin,size)
gr = GraphicShape.Make(arect)
gr.GetSymbol.SetStyle(#RASTERFILL_STYLE_SOLID)
gr.GetSymbol.SetColor(Color.GetWhite)
gra_list.Add(gr)
gra_list.MoveSelectedToFront
gr.SetSelected(TRUE)
gra_list.GroupSelected
'Move legend to lower-left corner of View
alegend = gra_list.GetSelected.Get(0)
alegend.Invalidate
alegend.SetOrigin(theView.GetDisplay.ReturnVisExtent.ReturnOrigin)
alegend.Invalidate
'Merge the graphic legend into the theme's graphic list
theTheme.GetGraphics.Merge(theView.GetGraphics.GetSelected)
gra_list.UnselectAll
av.ClearMsg
av.ClearStatus
```

movie2 **Displays charts in a sentence, category: tools, property: apply (script qmchar1u is used in the property: Update)**

```
'movie2.ave
theView = av.GetActiveDoc
theThemes = theView.GetActiveThemes
p = theView.GetDisplay.ReturnUserPoint
t = thethemes.get(0)
if (t.CanSelect.not) then
  exit
end
t.SelectByPoint(p,#VTAB_SELTYPE_NEW )
recs = t.FindByPoint(p)
if(recs.count = 0) then
  exit
end
vv = t.getftab.updateselection
selchart = {"Horizontal", "Vertical"}
c = msgbox.choiceasstring(selchart, "Make a selection",
    "Chart type selection")
if (c = nil) then
  exit
end
theView = av.GetActiveDoc
visnm = theView.GetVisibleThemes.get(0).getname
if ( visnm = "Gsflow") then
     prefix = "M"
     xnote = "Available: 1940 - 1991"
     if(c = "Horizontal") then
       TheChart = av.getproject.FindDoc("MovieMh")
     end
     if(c = "Vertical") then
       TheChart = av.getproject.FindDoc("MovieMv")
    end
end
if ( visnm = "unflow") then
     prefix = "Qm"
     xnote = "Available: 1960 - 1991"
     if(c = "Horizontal") then
       TheChart = av.getproject.FindDoc("MovieQmh")
     end
     if(c = "Vertical") then
       TheChart = av.getproject.FindDoc("MovieQmv")
    end
end
CHtable = TheChart.GetVtab
sp = "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
ln4 = "First year:"
ln5 = "Last year:"
labls = {sp, ln4,ln5}
fy = 1960
```

```
ty = 1991
fm = 1
tm = 12
defts = {" ", fy.asstring, ty.asstring}
while (true)
  xlist = msgBox.Multiinput(xnote,"Movie2", labls, defts )
  if (xlist.count = 0) then
    exit
  end
  if ((xlist.get(1).isnumber.not) or (xlist.get(1) < 1940)) then
    msgbox.error(xlist.get(1)++"is not a correct year", "Error in
field #1")
    continue
  end
  if ((xlist.get(2).isnumber.not) or (xlist.get(2) > 1991)) then
    msgbox.error(xlist.get(2)++"is not a correct year", "Error in
field #2")
    continue
  end
  break
end
av.showmsg(" Looking for the maximum value to set Y scale . . .")
fy = xlist.get(1).asnumber
ty = xlist.get(2).asnumber
bmap = Chtable.getselection
rec = -1
rec = bmap.getnextset(rec)
step = -1
lablst = list.make
'for each i in 0..11
'  lablst.add("x")
'end
'======== find maximum value  to set Y(or X) scale ========
xxmax = 0
for each yr in fy..ty
 for each mt in 1..12
  if (( yr = fy) and ( mt < fm)) then
    continue
  end
  if (( yr = ty) and ( mt > tm)) then
    continue
  end
  step = step + 1
  'if ( step < 12 ) then
  '  continue
  'end
  if (mt < 10 ) then
    txt = "0" + mt.AsString
  else
    txt = mt.AsString
  end
  name = prefix + yr.AsString + txt
  XFld=Chtable.FindField(name)
  xx22 = Chtable.returnvalueNumber(xfld,rec)
  if ( xx22 > xxmax ) then
```

```
    xxmax = xx22
  end
 end
end
nomonths = step
msgbox.info("Maximum value is"++xxmax.asstring,"Maximum")
av.clearmsg
av.showstopbutton
if(thechart.getwin.isopen.not) then
  thechart.getwin.open
  thechart.getwin.minimize
end
TheList=TheChart.GetFields
TheFList = TheChart.getVtab.GetFields
if(c = "Horizontal") then
  thechart.getYaxis.setboundsmax(xxmax)
end
if(c = "Vertical") then
  thechart.getXaxis.setboundsmax(xxmax)
end
step = -1
for each yr in fy..ty
av.showmsg(" YEAR: "++yr.asstring)
 for each mt in 1..12
  if (( yr = fy) and ( mt < fm)) then
    continue
  end
  if (( yr = ty) and ( mt > tm)) then
    continue
  end
  step = step + 1
  if ( step = 12 ) then
    if(msgbox.miniYesNo("Start ?", True).not) then
      exit
    end
    thechart.getwin.restore
    if(thechart.getwin.isopen.not) then
      thechart.getwin.activate
    end
  end
  if (mt < 10 ) then
    txt = "0" + mt.AsString
  else
    txt = mt.AsString
  end
  xstatus = av.setstatus(100 * step / nomonths)
  if(xstatus.not) then
     rlt = msgbox.yesno("Do you really want to stop?","Stop movie",
true)
     if(rlt) then
       rlt2 = msgbox.yesno("Do you want to minimize the chart
window?",
           "Close Window ?", true)
       if(rlt2) then
         thechart.getwin.minimize
       end
       exit
     else
       av.ClearStatus
       av.showstopbutton
     end
  end
  name = prefix + yr.AsString + txt
  NewTitle = txt + " / " + yr.asstring
  NewFld=TheChart.getVtab.FindField(name)
  TheList.Remove(0)
  Thelist.add(newfld)
  TheChart.GetTitle.SetName(NewTitle)
  theChart.SetSeriesFromRecords(true)
  theChart.Getwin.Open
 end
end
av.ClearStatus
rlt2 = msgbox.yesno("Do you want to minimize the chart?",
           "Close Window ?", true)
       if(rlt2) then
         thechart.getwin.minimize
       end
```

aliaset Sets the alias names for fields, that describe a chart axis.

```
'aliaset.ave
xFtable = av.GetActiveDoc.getvtab
prefix = "QM"
fy = 1960
ty = 1992
fm = 1
tm = 9
for each yr in fy..ty
 for each mt in 1..12
  if (( yr = fy) and ( mt < fm)) then
    continue
  end
  if (( yr = ty) and ( mt > tm)) then
    continue
  end
  if (mt < 10 ) then
    txt = "0" + mt.AsString
  else
    txt = mt.AsString
  end
  name = prefix + yr.AsString + txt
  XFld=xFtable.FindField(name)
txalias = mt.asstring+"/"+yr.asstring
xxx = XFld.setalias(txalias)
 end
end
```

B5 Scripts from the project “tools”

order6 Determines the order of a stream/watershed in a flow system

```
'order6.ave  Adds a field that contains a numbering
'            system to specify the stream order
'            in a flow system.
TBunit = av.GetActiveDoc
xxx = TBunit.getGUI
if ( xxx <> "Table") then
  xxx2 = TBunit.asstring++"is active"
  msgbox.warning("A Table must be active", xxx2)
  exit
end
VTunit = TBunit.getVtab
'assume the default fields in UNIT table:
TXunit = "Unit"
TXnext = "Downstream"
FDunit = VTunit.FindField(TXunit)
FDnext = VTunit.FindField(TXnext)
'create list of fields that are in VTunit
LSFDunit = VTunit.GetFields
'if the default fields can not be found, ask user for required
fields:
if(FDunit=nil) then
  'Select current_unit id field
  FDunit = MsgBox.List(LSFDunit, "Select a field that contains"
           ++NL++" the watershed/stream ID",
            "Unit_id Selection")
end
'if not selected, quit
IF ( FDunit = nil) then
  exit
end
if(FDnext=nil) then
  'Select downstream_unit id field
  FDnext = MsgBox.List(LSFDunit, "Select a field that contains"++NL++
             "the downstream unit ID", "Next_id Selection")
end
IF ( FDnext = nil) then
  exit
end
TXorderField = msgbox.input("Enter the name of order field ",
"Order Field Selection", "Order")
FDorder = vtunit.FindField(TXorderField)
xx = false
if (FDorder = nil) then
  xx = msgbox.YesNo( TXorderField++"field is not in the table "
  ++TBunit.asString+nl+"Do you want to create this field ?",
   "Field not Found !", true)
'end
  if (xx.not) then
```

```
    exit
  end
 'check if the VTunit can be edited if not, set it editable
 if(VTunit.isEditable.not) then
  VTunit.SetEditable(true)
 end
 if(VTunit.CanAddFields) then
   f1  =  Field.Make( TXorderfield,#FIELD_SHORT, 5, 0 )
   'Add fields
   VTunit.AddFields({f1})
 else
  msgBox.warning("Can not add a field to"++VTunit.asString,
                 "Can not edit table")
  exit
 end
else
 f1 = FDorder
 if(VTunit.isEditable.not) then
  VTunit.SetEditable(true)
 end
end
LSunit = list.make
LSnext = list.make
LSord = list.make
LSrecnxt = list.make
nx = VTunit.GetNumRecords
nn = nx - 1
nst = 100/nx   'variable used in status displaying
av.ShowMsg("Creating lists ...")
for each rec in VTunit
  av.ShowMsg("Creating lists ..."++rec.AsString)
  av.SetStatus(rec * nst)
  LSnext.Add(VTunit.ReturnValueNumber(FDnext, rec))
  LSunit.Add(VTunit.ReturnValueNumber(FDunit, rec))
  LSord.Add(1)
  LSrecnxt.Add(nx)
end
LSrecnxt.Add(nx)
'search for first order streams/watersheds and create list of the
'record numbers pointed by the "next/downstream unit" field
for each i in 0..nn
  av.ShowMsg("Searching for first order streams/watersheds
..."++i.asString)
  av.SetStatus(i * nst)
  for each j in 0..nn
    if ( LSnext.get(i) = LSunit.get(j) ) then
      LSord.set(j , 0 )
      LSrecnxt.set(i , j)
      Break
    end
  end
end
'write downstream record pointers into field f2
'for each rec in VTunit
'  VTunit.SetValue( f2, rec, LSrecnxt.Get( rec ))
```

```
'end
'determine order of the rest of streams
for each i in 0..nn
  av.ShowMsg("Calculating order ..."++i.asString)
  av.SetStatus(i * nst)
  if ( LSord.get(i) <> 1 ) then
    continue
  end
  j = LSrecnxt.get(i)
  k = 2
  while (j <> nx)
    if ( LSord.get(j) >= k ) then
      break
    end
    LSord.set(j , k )
    j = LSrecnxt.get(j)
    k = k + 1
  end 'while
end 'i
'write stream/watershed order into Table
for each rec in VTunit
  VTunit.SetValue( f1, rec, LSord.Get( rec ))
end
'refresh table and set it "not editable"
VTunit.Refresh
if(VTunit.isEditable) then
    VTunit.SetEditable(false)
end
```

upavg2 calculates weighted average over upstream units

```
' upavg2.ave calculates weighted average
'    for upstream units:
'    output = upstr_sum(input*wght)/upstr_sum(wght)
'=================================================
' note: dbf table contains a dummy (first) record
' that can influence calculations !!!
TBunit = av.GetActiveDoc
xxx = TBunit.getGUI
if ( xxx <> "Table") then
xxx2 = TBunit.asstring++"is active"
msgbox.warning("A Table must be active", xxx2)
exit
end
TheFtab = TBunit.getVtab
' Ask user for required fields:
TXunit = "unit_id"
TXnext = "unit_nx"
TXorder = "order"
TXitem = "input"
TXadd = "output"
TXwgt = "weight"
FDunit = theFtab.FindField(TXunit)
```

```
FDnext = theFtab.FindField(TXnext)
FDorder = theFtab.FindField(TXorder)
FDadd = theFtab.FindField(TXadd)
FDitem = theFtab.FindField(TXitem)
FDarea = theFtab.FindField(TXwgt)
' Make a list of VTunit fields
LSFDunit = theFtab.GetFields
' If the default fields can not be found, ask user
' for the required fields:
if(FDunit=nil) then
  ' Select current_unit id field (from entity)
  FDunit = MsgBox.List(LSFDunit, "Select a field that contains"
      ++NL++"the watershed/stream ID", "From-unit Selection")
end
' If not selected, quit
IF (FDunit = nil) then
  exit
end
if(FDnext=nil) then
  ' Select downstream_unit id field
  FDnext = MsgBox.List(LSFDunit, "Select a field that contains"
       ++NL++" the downstream unit ID", "To-unit Selection")
end
' If not selected, quit
IF ( FDnext = nil) then
  exit
end
if(FDorder=nil) then
  ' Select order field
  FDorder = MsgBox.List(LSFDunit, "Select field that contains"
     ++NL++" the watershed/stream order","Order field Selection")
end
' If not selected, quit
IF ( FDorder = nil) then
  xxx = msgbox.info("Create a field of stream order",
       "Missing fields")
  exit
end
if(FDitem=nil) then
  ' Select field that contains values to be averaged
  FDitem = MsgBox.List(LSFDunit, "Select a field that contains"
  ++NL++"values to be averaged", "Feature Selection")
end
' If not selected, quit
IF ( FDitem = nil) then
  exit
end
if(FDarea=nil) then
  ' Select field that contains weight
  FDarea = MsgBox.List(LSFDunit, "Select a field that contains"
  ++NL++"weight", "Weight Selection")
end
' If not selected, quit
IF ( FDarea = nil) then
  exit
```

```
end
' Prompt for a name (name of the field, that will be created)
TXadd = msgbox.input("Enter the name of output field ",
"Output Field Selection", "output")
if (TXadd = nil) then
  exit
end
FDadd = theFtab.FindField(TXadd)
xx = false
if (FDadd = nil) then
  xx = msgbox.YesNo( TXadd++"field is not in the table "
  ++TBunit.asString+nl+"Do you want to create this field ?",
   "Field not Found !", true)
'end
  if (xx.not) then
    exit
  end
  'check the setting of "input" field
  fenum = FDitem.gettype
  fprec = FDitem.getprecision
  fwidth = FDitem.getwidth
  xxx1 = msgbox.input("Please specify precision"++
    "(decimal part) of the output field",
    "Width of the new field" , fprec.asstring)
   fprec = xxx1.asnumber
  IF ( fprec = nil) then
    exit
  end
 xxx2 = msgbox.input("Please specify total width of the output
field",
   "Width of the new field" , fwidth.asstring)
   fwidth = xxx2.asnumber
  IF ( fwidth = nil) then
    exit
  end
  'check if the theFtab can be edited if not, set it editable
  if(theFtab.isEditable.not) then
    theFtab.SetEditable(true)
  end
  ' Create a new field
  if(theFtab.CanAddFields) then
    FDadd = field.Make(TXadd,fenum,fwidth,fprec)
    theFtab.AddFields({FDadd})
  else
    msgBox.warning("Can not add a field to"++theFtab.asString,
                   "Can not edit table")
    theFtab.SetEditable(false)
    exit
  end
else
  if(theFtab.isEditable.not) then
    theFtab.SetEditable(true)
  end
end
'create lists
```

```
ltunit = list.make
ltnext = list.make
ltorder = list.make
ltitem = list.make
ltadd = list.make
unnx = theFTab.GetNumRecords
unnn = unnx - 1
dcarea = dictionary.make(unnx)
dcmass = dictionary.make(unnx)
for each rec in theFTab
  Nunit = theFTab.ReturnValueNumber(FDunit, rec)
  ltunit.Add(Nunit)
  ltnext.Add(theFTab.ReturnValueNumber(FDnext, rec))
  ltorder.Add(theFTab.ReturnValueNumber(FDorder, rec))
  Narea = theFTab.ReturnValueNumber(FDarea, rec)
  if (Narea = 0 ) then
    msgbox.error("The weight can not be zero !"++
     NL++"record:"++rec.asstring++"field:"++
     FDarea.asstring, "Division by zero" )
     theFtab.SetEditable(false)
     exit
  end
  Nmass = theFTab.ReturnValueNumber(FDitem, rec) * Narea
  dcarea.add(Nunit, Narea )
  dcmass.add(Nunit, Nmass )
end
' find maximum order
unmaxord = 1
for each i in 0..unnn
  av.ShowMsg("Finding maximum order ..."++i.asString)
  istatus = i * 100 / unnn
  av.SetStatus(istatus)
  if (ltorder.get(i) > unmaxord ) then
        unmaxord = ltorder.get(i)
  end
end
'calculate cumulative values
unmaxord1 = unmaxord - 1
for each k in 1..unmaxord1
  av.ShowMsg("Calculating cumulative values ..."++k.asString)
  istatus = k * 100 / unmaxord
  av.SetStatus(istatus)
  for each i in 0..unnn
    if ( ltorder.get(i) <> k ) then
      continue
    end
    Nunit = ltunit.get(i)
    Nnext = ltnext.get(i)
    Nxmass = dcmass.get(Nunit) + dcmass.get(Nnext)
    Nxarea = dcarea.get(Nunit) + dcarea.get(Nnext)
    dcmass.set(Nnext, Nxmass)
    dcarea.set(Nnext, Nxarea)
  end
end
av.clearMsg
av.clearstatus
'write results to Ftab
for each rec in theFtab
  Nunit = theFTab.ReturnValueNumber(FDunit, rec)
  Xavg = dcmass.get(Nunit) / dcarea.get(Nunit)
  TheFtab.SetValue( FDadd, rec, Xavg )
end
theFtab.Refresh
if(TheFtab.isEditable) then
    theFtab.SetEditable(false)
end
```

cumul2 Accumulates values, going downstream (flow accumulation)

```
' cumul2.ave Adds a field or writes to the existing
'            field the accumulated values. Summation
'            is performed downstream, along the flow
'            path.
' note: dbf table contains a dummy (first) record
' that can influence calculations !!!
TBunit = av.GetActiveDoc
xxx = TBunit.getGUI
if ( xxx <> "Table") then
xxx2 = TBunit.asstring++"is active"
msgbox.warning("A Table must be active", xxx2)
exit
end
TheFtab = TBunit.getVtab
' Ask user for needed fields:
TXunit = "unit_id"
TXnext = "unit_nx"
TXorder = "order"
TXitem = "item_x"
FDunit = theFtab.FindField(TXunit)
FDnext = theFtab.FindField(TXnext)
FDorder = theFtab.FindField(TXorder)
'FDadd = theFtab.FindField(TXadd)
FDitem = theFtab.FindField(TXitem)
' Make a list of VTunit fields
LSFDunit = theFtab.GetFields
' If the default fields can not be found, ask user for
' the required fields:
if(FDunit=nil) then
  ' Select current_unit id field (from entity)
  FDunit = MsgBox.List(LSFDunit, "Select a field that contains"
      ++NL++"the watershed/stream ID", "From-unit Selection")
end
' If not selected, quit
IF (FDunit = nil) then
  exit
end
if(FDnext=nil) then
  ' Select downstream_unit id field
  FDnext = MsgBox.List(LSFDunit, "Select a field that contains"
       ++NL++" the downstream unit ID", "To-unit Selection")
end
' If not selected, quit
IF ( FDnext = nil) then
  exit
end
if(FDorder=nil) then
  ' Select order field
  FDorder = MsgBox.List(LSFDunit, "Select field that contains"
     ++NL++" the watershed/stream order","Order field Selection")
end
' If not selected, quit
IF ( FDorder = nil) then
  xxx = msgbox.info("Create a field of stream order",
       "Missing fields")
  exit
end
if(FDitem=nil) then
  ' Select field that contains values to be accumulated
  FDitem = MsgBox.List(LSFDunit, "Select a field that contains"
  ++NL++"values to be summed", "Feature Selection")
end
' If not selected, quit
IF ( FDitem = nil) then
  exit
end
' Prompt for a name (name of the field, that will be created)
TXadd = msgbox.input("Enter the name of output field ",
"Output Field Selection", "Accumulated")
FDadd = theFtab.FindField(TXadd)
xx = false
if (FDadd = nil) then
  xx = msgbox.YesNo( TXadd++"field is not in the table "
  ++TBunit.asString+nl+"Do you want to create this field ?",
   "Field not Found !", true)
'end
  if (xx.not) then
    exit
  end
  'check the setting of "item" field
  'ESRI gives no information how wide a field can be!
  fenum = FDitem.gettype
  fprec = FDitem.getprecision
  fwidth = FDitem.getwidth
  if(fwidth < 13) then
      fwidth = fwidth + 3
  elseif (fwidth = 13) then
    fwidth = 15
  elseif (fwidth > 13) then
  xxx = msgbox.input("Please specify a width of output field",
   "Width of the new field" , fwidth.asstring)
    fwidth = xxx.asnumber
  end
  IF ( fwidth = nil) then
    exit
  end
  'check if the theFtab can be edited if not, set it editable
  if(theFtab.isEditable.not) then
    theFtab.SetEditable(true)
  end
  ' Create a new field
  if(theFtab.CanAddFields) then
    FDadd = field.Make(TXadd,fenum,fwidth,fprec)
    theFtab.AddFields({FDadd})
  else
   msgBox.warning("Can not add a field to"++theFtab.asString,
                   "Can not edit table")
    exit
  end
else
  if(theFtab.isEditable.not) then
    theFtab.SetEditable(true)
  end
end
'create lists
ltunit = list.make
ltnext = list.make
ltorder = list.make
ltitem = list.make
ltadd = list.make
unnx = theFTab.GetNumRecords
unnn = unnx - 1
dcadd = dictionary.make(unnx)
for each rec in theFTab
  Nunit = theFTab.ReturnValueNumber(FDunit, rec)
  ltunit.Add(Nunit)
  ltnext.Add(theFTab.ReturnValueNumber(FDnext, rec))
  ltorder.Add(theFTab.ReturnValueNumber(FDorder, rec))
  Nitem = theFTab.ReturnValueNumber(FDitem, rec)
'  ltitem.Add(Nitem)
  dcadd.add(Nunit, Nitem )
end
' find maximum order
unmaxord = 1
for each i in 0..unnn
  av.ShowMsg("Finding maximum order ..."++i.asString)
  istatus = i * 100 / unnn
  av.SetStatus(istatus)
  if (ltorder.get(i) > unmaxord ) then
        unmaxord = ltorder.get(i)
  end
end
'calculate cumulative values
unmaxord1 = unmaxord - 1
for each k in 1..unmaxord1
  av.ShowMsg("Calculating cumulative values ..."++k.asString)
  istatus = k * 100 / unmaxord
  av.SetStatus(istatus)
  for each i in 0..unnn
    if ( ltorder.get(i) <> k ) then
      continue
    end
    Nunit = ltunit.get(i)
    Nnext = ltnext.get(i)
    Nxvalue = dcadd.get(Nunit) + dcadd.get(Nnext)
    dcadd.set(Nnext, Nxvalue)
  end
end
av.clearMsg
av.clearStatus
'write results to Ftab
for each rec in theFtab
  Nunit = theFTab.ReturnValueNumber(FDunit, rec)
  TheFtab.SetValue( FDadd, rec, dcadd.get(Nunit) )
end
theFtab.Refresh
if(TheFtab.isEditable) then
    theFtab.SetEditable(false)
end
```

decom2 calculates the difference between the inflows and the outflow, the reverse process to the flow accumulation

```
' decom2.ave for all modeling units calculates
' the difference between sum of inputs and the
' output ( deaccumulate values)
' note: dbf table contains a dummy (first) record
' that can influence the calculations !!!
TBunit = av.GetActiveDoc
xxx = TBunit.getGUI
if ( xxx <> "Table") then
xxx2 = TBunit.asstring++"is active"
msgbox.warning("A Table must be active", xxx2)
exit
end
TheFtab = TBunit.getVtab
' Ask user for requireded fields
TXunit = "unit_id"
TXnext = "unit_nx"
TXorder = "order"
TXitem = "input"
'TXadd = "decom_fd"
FDunit = theFtab.FindField(TXunit)
FDnext = theFtab.FindField(TXnext)
FDorder = theFtab.FindField(TXorder)
'FDadd = theFtab.FindField(TXadd)
FDitem = theFtab.FindField(TXitem)
' Make a list of VTunit fields
LSFDunit = theFtab.GetFields
' If the default fields can not be found, ask user for
' the required fields:
if(FDunit=nil) then
  ' Select current_unit id field (from entity)
  FDunit = MsgBox.List(LSFDunit, "Select a field that contains"
      ++NL++"the watershed/stream ID", "From-unit Selection")
end
' If not selected, quit
IF (FDunit = nil) then
  exit
end
if(FDnext=nil) then
  ' Select downstream_unit id field
  FDnext = MsgBox.List(LSFDunit, "Select a field that contains"
       ++NL++" the downstream unit ID", "To-unit Selection")
end
' If not selected, quit
IF ( FDnext = nil) then
  exit
end
if(FDorder=nil) then
  ' Select order field
  FDorder = MsgBox.List(LSFDunit, "Select field that contains"
     ++NL++" the watershed/stream order","Order field Selection")
end
' If not selected, quit
IF ( FDorder = nil) then
  xxx = msgbox.info("Create a field of stream order",
       "Missing fields")
  exit
end
if(FDitem=nil) then
  ' Select a field that contains values to be decomposed
  FDitem = MsgBox.List(LSFDunit,
          "Select a field that contains"++NL++
          "the values to be decomposed",
          "Input Field Selection")
end
' If not selected, quit
IF ( FDitem = nil) then
  exit
end
' Prompt for a name (name of the field, that will be created)
TXadd = msgbox.input("Enter the name of the output field ",
"Output Field Selection", "Decomposed")
FDadd = theFtab.FindField(TXadd)
xx = false
if (FDadd = nil) then
  xx = msgbox.YesNo( TXadd++"field is not in the table "
  ++TBunit.asString+nl+"Do you want to create this field ?",
   "Field not Found !", true)
'end
  if (xx.not) then
    exit
  end
  'check the setting of "item" field
  'width and the precision of the input field are assumed
  fenum = FDitem.gettype
  fprec = FDitem.getprecision
  fwidth = FDitem.getwidth
  ' Create a new field
  if(theFtab.isEditable.not) then
    theFtab.SetEditable(true)
  end
  if(theFtab.CanAddFields) then
    FDadd = field.Make(TXadd,fenum,fwidth,fprec )
    theFtab.AddFields({FDadd})
  else
  msgBox.warning("Can not add a field to"
      ++theFtab.asString, "Can not edit table")
  exit
  end
else
  if(theFtab.isEditable.not) then
    theFtab.SetEditable(true)
  end
end
'create lists
ltunit = list.make
ltnext = list.make
ltorder = list.make
ltitem = list.make
ltadd = list.make
unnx = theFTab.GetNumRecords
unnn = unnx - 1
dcadd = dictionary.make(unnx)
dcdec = dictionary.make(unnx)
for each rec in theFTab
  Nunit = theFTab.ReturnValueNumber(FDunit, rec)
  ltunit.Add(Nunit)
  ltnext.Add(theFTab.ReturnValueNumber(FDnext, rec))
  ltorder.Add(theFTab.ReturnValueNumber(FDorder, rec))
  Nitem = theFTab.ReturnValueNumber(FDitem, rec)
  dcadd.add(Nunit, Nitem )
  dcdec.add(Nunit, Nitem )
end
' find maximum order
unmaxord = 1
for each i in 0..unnn
  av.ShowMsg("Finding maximum order ..."++i.asString)
  istatus = i * 100 / unnn
  av.SetStatus(istatus)
  if (ltorder.get(i) > unmaxord ) then
        unmaxord = ltorder.get(i)
  end
end
'calculate decomposed values
unmaxord1 = unmaxord - 1
for each k in 1..unmaxord1
  av.ShowMsg("Decumulating values ..."++k.asString)
  istatus = k * 100 / unmaxord
  av.SetStatus(istatus)
  for each i in 0..unnn
    if ( ltorder.get(i) <> k ) then
```

```
      continue
    end
    Nunit = ltunit.get(i)
    Nnext = ltnext.get(i)
    Nxvalue = dcdec.get(Nnext) - dcadd.get(Nunit)
    dcdec.set(Nnext, Nxvalue)
  end
end
av.clearMsg
av.clearstatus
'write results to Ftab
for each rec in theFtab
  Nunit = theFTab.ReturnValueNumber(FDunit, rec)
  TheFtab.SetValue( FDadd, rec, dcdec.get(Nunit) )
end
theFtab.Refresh
if(TheFtab.isEditable) then
    theFtab.SetEditable(false)
end
```

Appendix C Arc/Info macros--AMLs

This Appendix contains the following Arc/Info procedures written in Arc/Info Macro Language (AML):

1) WSHGS.AML - Creates the following grids: stream network, watersheds, and watershed outlets;
2) NEXTWSH.AML - Creates an info file that contains two items: ID and the downstream watershed ID. Also creates a grid of watershed outlets that contains the ID of downstream watershed;
3) RAININFO.AML- Based on the time series stored in the attribute table of the point coverage, and the grid that specify the spatial division into zones, this AML calculates zonal averages and stores them in Arc/Info INFO table;
4) RAINMAP.AML - Creates grids of average monthly precipitation;
5) RAINM2.AML - Creates an INFO table from the grids of average monthly precipitation;
6) SELDATA.AML - Creates point coverage of stations with complete record of flow rate/precipitation depth in selected months;
7) FD4Y.AML - Supports the process of redistribution of the flow rate recorded in theUSGS stations;
8) SLOPE3.AML - Calculates the slope along a flow path;
9) DEMALB - Converts a grid into Albers Equal Area coordinate system;

C1 WSHGS.AML

```
/* -----------------------------------------------------------------
/*   Program: WSHGS.AML
/*   Purpose: Creates the following grids: stream network,
/*            watersheds, and watershed outlets.
/*            The watershed outlet grid contains:
/*            the most upstream cell of the first order stream,
/*            the most downstream cell of each reach, and the cells
/*            created from a point coverage of USGS stations.
/*            Runs only in GRID
/* -----------------------------------------------------------------
/*     Usage: &r WSHINFO <fdir> <facc> <fbas> <trsh> <fout> <item>
/*                   <fwsh> <flpp> <fstr>
/* Arguments: (input)
/*            fdir = (grid) flow direction
/*            facc = (grid) flow accumulation
/*            fbas = (grid) area under investigation (basin, mask)
/*            trsh = (value) threshold for stream delineation
/*                   ( if grids are in meters, then trsh is in km2 )
/*            fout = (point) selected flow gauging stations
/*               (additional pour points for watersheds delineation)
/*            item = (text) name of the item from <fout> which will
/*                   be used in "pointgrid" conversion.
/*
/*            (output, grids that will be created)
/*            fstr = (grid) stream network
/*            fwsh = (grid) drainage areas
/*            flpp = (grid) outlets
/*
/* Temporary: lso, lsl, lmx, lpd, lmn, lpu, lgs, lpx, (grids)
/*   Globals: none (see Temporary)
/*   Locals:  tresh (variable)
/* -----------------------------------------------------------------
/*     Notes: The first order watersheds (and the respective outlets)
/*            have ID number in the range 100000 < ID < 200000.
/*            Cellsize and Window of <fdir> are assumed.
/*            This AML checks neither for the existence and
/*            correctness of the input files nor for the existence
/*            of files that have the same names as the temporary
/*             files and the files to be created.
/*    If an error occurs then all grids and all info files that
/*    have the same name as output grids and temporary grids are
/     erased !!!
/* ----------------------------------------------------------------
/*   History: Author Pawel Mizgalewicz  11/21/93
/*            updated for point coverage  05/15/95
/*            working name net4.aml ( tested for the Cedar River
/*            - Iowa River basin, grid = 3000^2 cells, basin =
/*            3.2*10e6 cells of size 100m*100m).
/*            edited 6/13/95, 6/21/95, 2/19/96
/*=================================================================
&args fdir facc fbas trsh fout item fwsh flpp fstr
&severity &error &routine bailout

&if [SHOW PROGRAM] <> GRID &then ; &return This only runs in GRID.
&if [null %fstr%] &then
 &do
  &call usage
  &return
 &end

&messages &off
/* GRID settings:
&s xxcell = [show setcell]
&s xxwindow = [show setwindow]
setwindow %fdir%
setcell %fdir%
/* ------------------------------------------------------------------
/* Creating a stream network
/* ------------------------------------------------------------------
/* calculate threshold in number of cells
&sv trsh = %trsh% * 1000000    /* conversion into meters (map units)
&sv tresh = %trsh% / ( $$cellsize * $$cellsize )
      &type stream delineation ... [date -vfull]
%fstr% = con ( %facc% >= %tresh% AND %fbas% > 0 , 1 )
/* ------------------------------------------------------------------
/* Finding the most downstream cell in each reach (watershed outlet)
/* ------------------------------------------------------------------
      &type streamorder [date -vfull]
lso = streamorder ( %fstr% , %fdir% )
      &type streamlink [date -vfull]
lsl = streamlink ( %fstr% , %fdir% )
      &type zonalmax [date -vfull]
lmx = zonalmax (lsl , %facc% )
      &type lpd condition [date -vfull]
lpd = con ( lmx == %facc% , lsl, 0 )
kill lmx all
/* lpd contains cells that have been selected by finding cell with
/* the largest value of the flowaccumulation within each partial
/*  drainage area.
/* lpd cells have value equal to the stream_ID and the watershed-ID.

/* ------------------------------------------------------------------
/* Finding the most upstream cell in each first order reach (wshed
outlet)
/* ------------------------------------------------------------------
      &type zonalmin [date -vfull]
lmn = zonalmin (lsl , %facc% )
      &type lpu condition [date -vfull]
lpu = con ( lso == 1 AND lmn == %facc%, lsl + 100000 , 0 )
      &type lfs focalsum for first order wshds [date -vfull]
kill lso all ; kill lsl all ; kill lmn all
/* lpu contains cells that have been selected by finding cell with
/* the smallest value of the flowaccumulation within first order
/* partial watersheds.
/* lpu cells have value equal to the stream_ID (and the watershed-ID)
/* increased by 100 000. Numbers >100 000 and <200 000 are used to
/* identify first order watersheds
/* (flow gauging station_ids are large numbers, for example 5448500)

/* -----------------------------------------------------------------
/* Converting point coverage into grid.
/* Cell value = value from %item%
/* -----------------------------------------------------------------
lgs = pointgrid ( %fout% , %item%, #, #, #, ZERO )

/* -----------------------------------------------------------------
/* Creating grid with all potential outlets
/* -----------------------------------------------------------------
      &type all outlets [date -vfull]
lpx = con ( lpu == 0 , con ( lgs == 0 , lpd, lgs ) , lpu )
kill lpu all    /* upper end of first order reach
kill lgs all    /* usgs gauging stations
kill lpd all    /* down (lowest) end of the reach
      &type setnull cells == 0  [date -vfull]
%flpp% = setnull ( lpx < 1, lpx )
&if not [ exists %flpp%.vat -info ] &then ; buildvat %flpp%
kill lpx all
/* all watershed outlets must be located on the stream network.
/* Gauging stations, which are located on the first cell of the first
/* order stream will not affect the value of the cell e.g. node
number
/* will not be changed.
/* -----------------------------------------------------------------
/* Delineating watersheds
/* -----------------------------------------------------------------
      &type watershed [date -vfull]
%fwsh% = watershed ( %fdir% , %flpp% )
&if not [exists %fwsh%.vat -info] &then ; buildvat %fwsh%
      &type watershed end [date -vfull]
/* restore GRID settings:
setcell %xxcell%
setwindow %xxwindow%
&messages &on
&return
/*-------------
&routine USAGE
/*-------------
&type Usage: &r WSHGS <fdir_grid> <facc_grid> <basin_grid>
<tresh_num>
&type              <fout_point> <item> <watershed> <pour_points>
<streams>
&return &inform
/*-------------
&routine EXIT
/*-------------
/* delete all temporary files:
&if [exists lso -grid ] &then ; kill lso all•
&if [exists lsl -grid ] &then ; kill lsl all
&if [exists lmx -grid ] &then ; kill lmx all
```

```
&if [exists lpd -grid ] &then ; kill lpd all
&if [exists lmn -grid ] &then ; kill lmn all
&if [exists lpu -grid ] &then ; kill lpu all
&if [exists lgs -grid ] &then ; kill lgs all
&if [exists lpx -grid ] &then ; kill lpx all
/* on error erase grids that exist or that have been created
&if [exists %fwsh% -grid ] &then ; kill %fwsh% all
&if [exists %flpp% -grid ] &then ; kill %flpp% all
&if [exists %fstr% -grid ] &then ; kill %fstr% all
/* restore GRID settings:
setcell %xxcell%
setwindow %xxwindow%
&messages &on
&return
/*--------------
&routine BAILOUT
/*--------------
&severity &error &ignore
&call exit
&return &warning An error has occurred in WSHGS.AML
/* -------------------------------------------------------------
/*     EXAMPLE OF APPLICATION
/* -------------------------------------------------------------
/* grid
/*      /* input
/* &s fdir = $HOME/iowa/data/crfdr     /* crfdr =
flowdirection(elev_grid)
/* &s facc = $HOME/iowa/data/crfac     /* crfac = flowaccumulation
(crfdr)
/* &s fbas = $HOME/iowa/data/crbas     /* basin or watershed
/* &s trsh = 25                        /* units in km2
/* &s fout = $home/iowa/test90/stations/gsflow  /* point coverage
/* &s item = station_id                /* item in %fout% coverage
/*      /* output
/* &s fwsh = crwsh                     /* watersheds
/* &s flpp = crlpp                     /* pour points
/* &s fstr = crstr                     /* stream network
/*
/* &r wshgs %fdir% %facc% %fbas% %trsh% %fout% %item% %fwsh% %flpp%
%fstr%
/*
/* q                                   /* quit from GRID
/*
/* &return                             /* return from example AML
/* -------------------------------------------------------------
```

C2 NEXTWSH.AML

```
/* -------------------------------------------------------------------
/*   Program: NEXTWSH.AML
/*   Purpose: Creates an info file that contains two items: ID and
/*            the downstream watershed ID. Also creates a grid of
/*            watershed outlets that contains the ID of downstream
/*            watershed.
/*            Runs only in GRID
/* -------------------------------------------------------------------
/* Usage: &r NEXTWSH <fdir> <fwsh> <flpp> <infn> <idin> <nxin> <fnxt>
/* Arguments: (input)
/*            fdir = (grid) flow direction
/*            fwsh = (grid) watersheds (partial drainage areas)
/*                          value in VAT = unit_id number
/*            flpp = (grid) pour points ( watershed outlets)
/*                          value in VAT = unit_id number
/*            (output)
/*            infn = (info file) name of the info file to be created
/*            idin = (item) name of the item in which the unit_id
/*                          number will be stored
/*            nxin = (item) name of the item in which the downstream
/*                          unit_id will be stored
/*            fnxt = (grid) similar to %flpp% (watershed outlets):
/*                     the values in VAT are not equal to the unit_id
/*                     number, they equal to the downstream unit
/*                     ID number.
/* Temporary: xxxtmp, xxxcomnx, (GRIDS)
/*            xxxtmp1 (INFO)
/*   Globals:
/*    Locals: see Temporary
/* ------------------------------------------------------------------
/*     Notes: The ID of the next watershed for the most downstream
/*            watershed equals 0.
/*            If the flowdirection in watershed pour point can't be
/*            determined, the next watershed ID = -1
/*        This AML checks neither for the existence and correctness
/*        of the input files nor for the existence of files that have
/*        the same names as the temporary files and the files to be
/*        created.
/*        If an error occurs then all grids and all info files that
/*        have the same name as output grids/info files and temporary
/*            grids/info are erased !!!
/* ---------------------------   --------------------------------
/*   History: coded by Pawel Mizgalewicz  12/20/93
/*         and converted into a stand alone procedure
/*         6/10 - 6/21/95, 2/19/96
/*  ==================================================================
&args fdir fwsh flpp infn idin nxin fnxt
&severity &error &routine bailout

&if [SHOW PROGRAM] <> GRID &then ; &return This only runs in GRID.
```

```
&if [null %fnxt%] &then
 &do
  &call usage
  &return
 &end

&messages &off
/* GRID settings:
&s xxcell = [show setcell]
&s xxwindow = [show setwindow]
setcell %fwsh%
setwindow %fwsh%
/* --------------------------------------------------------------------
&type searching for next (downstream) watershed ...   [date -vfull]
/* --------------------------------------------------------------------
&sv wsh = xxxtmp
%wsh% = con ( isnull ( %fwsh%), 0, %fwsh% )
%fnxt% = con (%flpp% > 0, con (%fdir% == 1, %wsh%(1,0), ~
        con (%fdir% == 2, %wsh%(1,1), ~
        con (%fdir% == 4, %wsh%(0,1), ~
        con (%fdir% == 8, %wsh%(-1,1), ~
        con (%fdir% == 16, %wsh%(-1,0), ~
        con (%fdir% == 32, %wsh%(-1,-1), ~
        con (%fdir% == 64, %wsh%(0,-1), ~
        con (%fdir% == 128, %wsh%(1,-1), -1)))))))))
kill xxxtmp all
&if not [exists %fnxt%.vat -info] &then ; buildvat %fnxt%
xxxcomnx = combine ( %flpp% , %fnxt% )
/* the %fnxt% grid is similar to the %flpp%, watershed outlets grid.
/* The cell value is equal to the next watershed ID number
/* --------------------------------------------------------------------
&type Creating the info file  ...  [date -vfull]
/* --------------------------------------------------------------------
arc additem xxxcomnx.vat xxxtmp1 %idin% 4 8 B
arc additem xxxtmp1 xxxtmp1 %nxin% 4 8 B
cursor xxnext declare xxxtmp1 info rw
cursor xxnext open
&do &while %:xxnext.aml$next%
  &s :xxnext.%idin% = [value :xxnext.%flpp%]
  &s :xxnext.%nxin% = [value :xxnext.%fnxt%]
  cursor xxnext next
&end
cursor xxnext close
cursor xxnext remove
arc pullitems xxxtmp1 %infn%
%idin%
%nxin%
end
kill xxxcomnx all
&s x = [delete xxxtmp1 -info]
/* restore GRID settings:
setcell %xxcell%
setwindow %xxwindow%
&messages &on
&return
```

```
/*-------------
&routine USAGE
/*-------------
&type Usage: &r NEXTWSH <fdir_grid> <wsh_grid> <pourpt_grid>
<newINF_name>
&type               <id_item_name> <next_id_item_name> <nextwsh_grid>
&return &inform
/*-------------
&routine EXIT
/*-------------
/* delete all temporary files:
&if [exists xxxtmp1 -info ] &then ; &s x = [delete xxxtmp1 -info]
&if [exists xxxtmp -GRID] &then ; kill xxxtmp all
&if [exists xxxcomnx -GRID] &then ; kill xxxcomnx all
&if [exists %fnxt% -GRID] &then ; kill %fnxt% all
/* restore GRID settings:
setcell %xxcell%
setwindow %xxwindow%
&messages &on
&return
/*--------------
&routine BAILOUT
/*--------------
&severity &error &ignore
&call exit
&return &warning An error has occurred in NEXTWSH.AML
/* ----------------------------------------------------------------
/*     EXAMPLE OF APPLICATION
/* ----------------------------------------------------------------
/* grid
/*      /* input
/* &s fdir = $HOME/iowa/data/crfdr/* crfdr = flowdirection(elev_grid)
/* &s fwsh = crwsh                /* watersheds
/* &s flpp = crlpp                /* pour points
/*     /* output
/* &s fnxt = crnxt                /* outflow,value=downstream wshd ID
/* &s infn = %fnxt%.dat           /* new info file
/* &s idin = unit_id              /* item (unit ID)
/* &s nxin = next_id              /* item ( downstream unit ID)
/*
/* &run nextwsh %fdir% %fwsh% %flpp% %infn% %idin% %nxin% %fnxt%
/*
/* q                              /* quit from GRID
/*
/* &return                        /* return from example AML
/* ----------------------------------------------------------------
```

C3 RAININFO.AML

```
/* -----------------------------------------------------------------
/*   Program: RAININFO.AML
/*   Purpose: Based on the time series stored in the attribute table
/*            of the point coverage, and the grid that specify the
/*            spatial division into zones, this AML calculates
/*            zonal averages and stores them in Arc/Info INFO table
/*   Method:  For a given month of the selected period this AML
/*            creates a grid of spatially distributed feature
/*            such as rainfall, temperature, or evaporation.
/*            The Inverse Distance Weighting IDW procedure is
/*            applied. To use different method only one line
/*            need to be changed. In the next step this AML
/*            calculates average values for all zones, specified by
/*            and stores them in INFO file. The procedure is
/*            repeated for all months from the specified time
/*            Runs only in GRID
/* -----------------------------------------------------------------
/*     Usage: &r RAININFO <fy> <fm> <ty> <tm> <p_data> <zone>
/*                       <outinf> {id_itm} {prc} {kcell} {prx}
/* Arguments: (input)
/*            fy, fm = from year, from month
/*            ty, tm = to year, to month
/*            p_data = (coverage) point coverage that contains time
/*                     series
/*            zone   = (grid) defines zones and their IDs
/*            outinf = (info) output info file to be created
/*            id_itm = (item) name of the item in which the zone ID
/*                     will be stored, default {id_itm} = zone_id
/*            prc    = (number) the values must be converted
/*                     into integer numbers to create VAT.
/*                     Before conversion they are multiplied by
/*                     <prc> to preserve decimal part of value.
/*                     default {prc} = 1000 (three decimal places)
/*            kcell  = (number) cell size multiplier.
/*                     The grid that is created from point coverage
/*                     will assume the size of the cell from
/*                     <zone> grid multiplied by {kcell}
/*                     default {kcell} = 1
/*            prx    = (prefix) Items that are created
/*                     in <outinf> info file will named as follows:
/*                     <prx> + item name from input coverage <p_data>
/*                     default {prx} = "p".
/* Temporary: xxx, xxx2, xxxcom (grid), xxxtmp1 (info)
/*    Locals: ab, bb, cc, nitem, prfnitem, yr,mt  (variables)
/* -----------------------------------------------------------------
/*     Notes: The following convention is used for naming the items
/*            in PAT (table that stores original time series)
/*            myyyymm,    where:  m indicates monthly values,
/*            yyyy = year, mm = month, for example item m197603
/*            carries data for March 1976.
/* -------------------------------------------------------------------
/*    History: Coded by Pawel Mizgalewicz  05/05/95
/*             Modified 06/18/1995 - 6/21/95,
/*             02,20,96 - defaults added for id_itm prc kcell.
/* -------------------------------------------------------------------

&args fy fm ty tm p_data zone outinf id_itm prc kcell prx

&severity &error &routine bailout

&if [SHOW PROGRAM] <> GRID &then ; &return This only runs in GRID.
&if [null %outinf%] &then
 &do
  &call usage
  &return
 &end
&if [null %prx%] &then ; &s prx = p
&if [null %kcell%] &then ; &s kcell = 1
&if [null %prc%] &then ; &s prc = 1000
&if [null %id_itm%] &then ; &s id_itm = zone_id

&messages &off
/* GRID settings:
&s xxcell = [show setcell]
&s xxwindow = [show setwindow]
&describe %zone%
&sv size = [calc %grd$dx% * %kcell%]
&type cell size = %size%
/* -------------------------------------------------------------------
/*       Make a copy of %zone%.VAT file and add item %zone%
/* -------------------------------------------------------------------
arc additem %zone%.vat xxxtmp1 %zone% 4 5 B
cursor xxcur1 declare xxxtmp1 info rw
cursor xxcur1 open
&do &while %:xxcur1.aml$next%
  &s :xxcur1.%zone% = %:xxcur1.VALUE%
  cursor xxcur1 next
&end
cursor xxcur1 close
cursor xxcur1 remove
/* -------------------------------------------------------------------
/*       Do it for all selected years and months
/* -------------------------------------------------------------------
&DO yr = %ty% &to %fy% &by -1
  &DO mt = 12 &to 1 &by -1
    &s ab = [calc %yr% = %fy%] AND [calc %mt% lt %fm%]
    &s bb = [calc %yr% = %ty%] AND [calc %mt% gt %tm%]
    &s cc = %ab% OR %bb%
    &IF NOT %cc%  &THEN
      &do
        &s templ = mxxxx0y
        &s nitem = [subst %templ% xxxx %yr% ]
        &if %mt% lt 10 &then
            &sv nitem = [subst %nitem% y %mt% ]
          &else
```

```
            &sv nitem = [subst %nitem% 0y %mt% ]
        &type  processing item %nitem% ... [date -vfull]
        setcell %size%
        setwindow MAXOF
        xxx = idw ( %p_data% , %nitem%  )
        setcell %zone%
        setwindow %zone%
        xxx2 = int ( %prc% * zonalmean ( %zone% , xxx  ))
        kill xxx all
        xxxcom = combine (%zone%, xxx2)
        kill xxx2 all
        &s prfnitem = %prx%%nitem%
        arc additem xxxcom.vat xxxcom.vat %prfnitem% 4 12 F 4 xxx2
        cursor xxcur1 declare xxxcom.vat info rw
        cursor xxcur1 open
        &do &while %:xxcur1.aml$next%
           &s :xxcur1.%prfnitem% = [calc %:xxcur1.xxx2% / %prc%]
           cursor xxcur1 next
        &end
        cursor xxcur1 close
        cursor xxcur1 remove
        arc joinitem xxxtmp1 xxxcom.vat xxxtmp1 %zone% %zone%
        arc dropitem xxxtmp1 xxxtmp1 xxx2
        kill xxxcom all
      &end
 &end
&end
/* ------------------------------------------------------------
&type Adding ID item ... [date -vfull]
/* ------------------------------------------------------------
arc additem xxxtmp1 xxxtmp1 %id_itm% 4 8 B # %zone%
cursor xxcur2 declare xxxtmp1 info rw
cursor xxcur2 open
&do &while %:xxcur2.aml$next%
           &s :xxcur2.%id_itm% = [value :xxcur2.%zone%]
           cursor xxcur2 next
&end
cursor xxcur2 close
cursor xxcur2 remove
/* ------------------------------------------------------------
&type Cleaning INFO table (dropping redundant items) .. [date -vfull]
/* ------------------------------------------------------------
arc dropitem xxxtmp1 xxxtmp1 value
arc dropitem xxxtmp1 xxxtmp1 count
arc dropitem xxxtmp1 %outinf% %zone%
&s x = [delete xxxtmp1 -info]
/* restore GRID settings:
setcell %xxcell%
setwindow %xxwindow%
&messages &on
&return
/*-------------
&routine USAGE
/*-------------
&type Usage: &r RAININFO <fy> <fm> <ty> <tm> <pnt_data> <zone>
```

```
&type                   <out_inf> <ID_item> <prc> <kcell> {prefix}
&return &inform
/*-------------
/* &routine CHECK
/*-------------
/* IF temporary file exists, inform and exit
/* If file to be build exists , inform and exit
/* If input file is not correct or does not exist, inform and exit
/* return
/*-------------
&routine EXIT
/*-------------
/* delete all temporary files:
&if [exists xxxtmp1 -info ] &then;   &s x = [delete xxxtmp1 -info]
&if [exists xxx -GRID] &then ; kill xxx all
&if [exists xxx2 -GRID] &then ; kill xxx2 all
&if [exists xxxcom -GRID] &then ; kill xxxcom all
/* restore GRID settings:
setcell %xxcell%
setwindow %xxwindow%
&messages &on
&return
/*--------------
&routine BAILOUT
/*--------------
&severity &error &ignore
&call exit
&return &warning An error has occurred in RAININFO.AML
/* ----------------------------------------------------------------
/*     EXAMPLE OF APPLICATION
/* ----------------------------------------------------------------
/* grid
/* &s fy = 1990
/* &s fm = 1
/* &s ty = 1990
/* &s tm = 12
/* &s pdat = ../stations/gsrain
/* &s zone = crwsh
/* &s id unit_id
/*
/* &r raininfo %fy% %fm% %ty% %tm% %pdat% %zone% prcinf2 %id% 10000
10 r
/*
/* q
/* &return
/* ----------------------------------------------------------------
```

C4 RAINMAP.AML

```
/* ------------------------------------------------------------------
/*   Program: RAINMAP.AML
/*   Purpose: Creates GRIDS of average monthly precipitation
/*            (10e-3 cm/d -- value from PAT is multiplied by 1000,
/*             assumed cell size = 1000)
/*            Stations that have complete record of precipitation for
/*            a given month (from specified time period) are used.
/*            rainm2.aml uses output to put values into an INFO file
/*            Runs only in GRID
/* ------------------------------------------------------------------
/*     Usage: &r RAINMAP <fy> <fm> <ty> <tm> <xxxdat> {YES|NO}
/* Arguments: (input)
/*            fy, fm = from year, from month
/*            ty, tm = to year, to month
/*            xxxdat = info file that contains precipitation depth
/*            yes|no = yes - include records with values = 0
/*                     no - exclude records and stations that have 0
/*                          (default = no)
/* Temporary: xxxxx1
/*    Locals: templ, nitem, nit, yr,mt, selec (variables)
/* ------------------------------------------------------------------
/*     Notes: Data are stored in PAT (point). Each record is related
/*            to gauging station. The following naming convention is
/*            used for items: myyyymm, where:
/*            m = indicates monthly values,
/*            yyyy = year, mm = month, for example, item m197603
/*            contains records for March 1976
/*            Info file is created by rainm2.aml
/* ------------------------------------------------------------------
/*    History: Coded by Pawel Mizgalewicz  08/08/95
/*             edited 02/20/96
/* ------------------------------------------------------------------
&args fy fm ty tm xxxdat zero

&severity &error &routine bailout

&if [SHOW PROGRAM] <> GRID &then ; &return This only runs in GRID.
&if [null %xxxdat%] &then
 &do
  &call usage
  &return
 &end
&if [null %zero%] &then ; &s zero = YES
&s xx = [translate %zero%]
&select %xx%
  &when NO
    &s klm = gt
  &when YES
    &s klm = ge
  &otherwise
```

```
    &do
      &call usage
      &return
    &end
&end /* select

&messages &off
setwindow maxoff
setcell 1000

&DO yr = %fy% &to %ty%
  &DO mt = 1 &to 12
    &s ab = [calc %yr% = %fy%] AND [calc %mt% lt %fm%]
    &s bb = [calc %yr% = %ty%] AND [calc %mt% gt %tm%]
    &s cc = %ab% OR %bb%
    &IF NOT %cc%  &THEN
      &do
        /* &s nit = %nit% + 1
        &s templ = mxxxx0y
        &s nitem = [subst %templ% xxxx %yr% ]
        &if %mt% lt 10 &then
             &sv nitem = [subst %nitem% y %mt% ]
           &else
             &sv nitem = [subst %nitem% 0y %mt% ]
       &type %nitem%
       /* &s namit%nit% = %nitem%
       &sv selec = res %nitem% %klm% 0
       &type selecting complete records %selec%

arc reselect %xxxdat% xxxxx1 point
%selec%
~
n
n

      &end

    arc build xxxxx1 point
    p%nitem% = int ( 1000 * idw ( xxxxx1, %nitem% ) )
    kill xxxxx1 all

  &end
&end
&messages &on

&return
/*-------------
&routine USAGE
/*-------------
&type Usage: &r RAINMAP <from_yr> <from_mt> <to_yr> <to_mt>
<data_point>
&type               {YES|NO}
&return &inform
/*-------------
&routine EXIT
```

```
/*-------------
/* delete all temporary files:
&if [exists xxxxx1 -COVER] &then ; kill xxxxx1 all
&messages &on
&return
/*--------------
&routine BAILOUT
/*--------------
&severity &error &ignore
&call exit
&return &warning An error has occurred in RAINMAP.AML
/* ------------------------------------------------------------
/*     EXAMPLE OF APPLICATION
/* ------------------------------------------------------------
/*      /* input
/* &s data = $HOME/iowa/data/gsprec    /* point coverage and data
/* grid
/* &run rainmap 1940 1 1993 6 %data% YES
/* q
/* &return                             /* return to Arc/Info prompt
/* ------------------------------------------------------------
```

C5 RAINM2.AML

```
/* ------------------------------------------------------------------
/*   Program: RAINM2.AML
/*   Purpose: Creates an INFO table from the grids of average
/*            monthly precipitation (needs grid of zones)
/*            (10e-3 cm/d -- value from PAT was multiplied by 1000,
/*             in original rainmap.aml assumed cell size = 1000 )
/*            (rainmap.aml creates grids of precipitation)
/*            Runs only in GRID, output: integer values !!!
/* ------------------------------------------------------------------
/*     Usage: &r RAINM2 <fy> <fm> <ty> <tm> <ppath> <zones> <idname>
/*                      <frominfo> <toinfo>
/* Arguments: (input)
/*            fy, fm = from year, from month
/*            ty, tm = to year, to month
/*            ppath = path to directory that contains grids of
/*                    precipitation
/*            zones = zones for calculatuing average prec. depth
/*            idname = same as zones, contains identification
/*                      number for zones (unit_id, modeling units)
/*            frominfo = info file that contains station id numbers
/*            toinfo = output info file
/*                     copy of frominfo + precipitation items
/* Temporary: xxxcom
/*    Locals: templ, nitem, nit, yr,mt,(variables)
/* ------------------------------------------------------------------
/*     Notes: Post processor for RAINMAP.AML
/*            RAINMAP.aml changes units, here means are converted to
/*            integer values !!!
/*            The following naming convention is
/*            used for items: myyyymm, where:
/*            m = indicates monthly values,
/*            yyyy = year, mm = month, for example, item m197603
/*            contains records for March 1976
/*            Info file is created by rainm2.aml
/* ------------------------------------------------------------------
/*    History: Coded by Pawel Mizgalewicz  95
/*             edited 02/20/96
/* ------------------------------------------------------------------
&args fy fm ty tm ppath zones idname frominfo toinfo

/* &s ppath = ../rain25/
/* &s zones = crwsh
/* &s idname = unit_id
/* &s frominfo = crwshc.dat
/* &s toinfo = crwshc.datp

/* &sv fy = 1990
/* &sv fm = 1
/* &sv ty = 1990
/* &sv tm = 12
```

```
copyinfo %frominfo% %toinfo%

&type ...aml, start  [date -vfull]
grid
&messages &off
setwindow %zones%
setcell %zones%

/* %idname% = %zones%
/* repeat in reverse order
&DO yr = %ty% &to %fy% &by -1
  &DO mt = 12 &to 1 &by -1
    &IF %mt% ge %fm% and %mt% le %tm% &THEN
      &do
        &s templ = mxxxx0y
        &s nitem = [subst %templ% xxxx %yr% ]
        &if %mt% lt 10 &then
             &sv nitem = [subst %nitem% y %mt% ]
           &else
             &sv nitem = [subst %nitem% 0y %mt% ]

&type p%nitem% processing ... [date -vfull]
p%nitem% = zonalmean ( %zones% , %ppath%p%nitem%
xxxcom = combine ( %idname% , p%nitem% )
kill p%nitem% all
&sys arc joinitem %toinfo% xxxcom.vat %toinfo% %idname% %idname%
kill xxxcom all

     &end
  &end
&end

&type rain series - point cover created, exiting ... [date -vfull]

&return
```

C6 SELDATA.AML

```
/* -------------------------------------------------------------------
/*   Program: SELDATA.AML
/*   Purpose: Selects the items that contain data for specified
/*            time period. Then, selects gauging stations that
/*            have complete record (no missing data).
/*            Creates point coverage of selected stations with data
/*            for specified months
/*            Runs only in ARC
/* -------------------------------------------------------------------
/*     Usage: &r SELDATA <xxxdat> <xxxhed> <joinit> <xxxout> {YES|NO}
/* Arguments: (input)
/*            fy, fm = from year, from month
/*            ty, tm = to year, to month
/*            xxxdat = info file that contains flow database
/*            xxxhed = point coverage that represents location of
/*                     observation stations. xxxdat and xxxhed must
/*                     contain common item to relate these tables
/*                    (for example item station_id)
/*            joinit = name of the common item for both tables
/*            (output)
/*            xxxout = point coverage which contains selected flow
/*                     records or precipitation measurements,
/*                     as well as selected gauging stations.
/*            yes|no = yes - include records with values = 0
/*                     no - exclude records and stations that have 0
/*                          (default = no)
/* Temporary: %xxxout%tmp (cover), xxxxx1 (info)
/*    Locals: temp1, nitem, nit, yr,mt, selec (variables)
/* -------------------------------------------------------------------
/*     Notes: Data file is an INFO file. Each record is related to
/*            gauging station. The following naming convention is
/*            used for items:
/*            myyyymm, where m indicates monthly values,
/*            yyyy = year, mm = month, for example item m197603
/*            contains records for March 1976
/*            Due to the Arc/Info restrictions only about 65-month
/*            period can be selected in single run of this AML
/* -------------------------------------------------------------------
/*    History: Coded by Pawel Mizgalewicz  05/05/95
/*             Modified 06/15/1995, edited 2/20/96
/* -------------------------------------------------------------------
&args fy fm ty tm xxxdat xxxhed joinit xxxout zero

&severity &error &routine bailout

&if [SHOW PROGRAM] <> ARC &then ; &return This only runs in ARC.
&if [null %xxxout%] &then
 &do
  &call usage
  &return
 &end
&if [null %zero%] &then ; &s zero = no
&s xx = [translate %zero%]
&select %xx%
  &when NO
    &s klm = gt
  &when YES
    &s klm = ge
  &otherwise
    &do
      &call usage
      &return
    &end
&end /* select

/* --------------------------------------------------------------------
/*       Make a copy of point coverage
/* --------------------------------------------------------------------
copy %xxxhed% %xxxout%tmp   /* create temporary file
&sv pultmp = xxxxx1         /* temporary info file
/* --------------------------------------------------------------------
&type Selecting years and months  ...         [date -vfull]
/* --------------------------------------------------------------------
PULLITEMS %xxxdat% %pultmp%
%joinit%
&DO yr = %fy% &to %ty%
  &DO mt = 1 &to 12
    &s ab = [calc %yr% = %fy%] AND [calc %mt% lt %fm%]•
    &s bb = [calc %yr% = %ty%] AND [calc %mt% gt %tm%]
    &s cc = %ab% OR %bb%
    &IF NOT %cc%  &THEN
      &do
        &s templ = mxxxx0y
        &s nitem = [subst %templ% xxxx %yr% ]
        &if %mt% lt 10 &then
            &sv nitem = [subst %nitem% y %mt% ]
          &else
            &sv nitem = [subst %nitem% 0y %mt% ]
        &type  selecting item %nitem%
%nitem%
      &end
 &end
&end

end

/* join selected items with header file (point coverage)
joinitem %xxxout%tmp.pat xxxxx1 %xxxout%tmp.pat %joinit% %joinit%
&sv st = [delete xxxxx1 -info]
/* --------------------------------------------------------------------
&type Selecting gauging stations that have full record  ...  [date -
vfull]
/* --------------------------------------------------------------------
reselect %xxxout%tmp %xxxout% point
&DO yr = %fy% &to %ty%
```

```
  &DO mt = 1 &to 12
    &s ab = [calc %yr% = %fy%] AND [calc %mt% lt %fm%]
    &s bb = [calc %yr% = %ty%] AND [calc %mt% gt %tm%]
    &s cc = %ab% OR %bb%
    &IF NOT %cc%  &THEN
      &do
        /* &s nit = %nit% + 1
        &s templ = mxxxx0y
        &s nitem = [subst %templ% xxxx %yr% ]
        &if %mt% lt 10 &then
             &sv nitem = [subst %nitem% y %mt% ]
           &else
             &sv nitem = [subst %nitem% 0y %mt% ]
       &type %nitem%
       /* &s namit%nit% = %nitem%
       &sv selec = res %nitem% %klm% 0
       &type selecting complete records %selec%
%selec%
~
n
y

      &end
  &end
&end
&type end
%selec%
~
n
n

build %xxxout% point
kill %xxxout%tmp all
&return
/*-------------
&routine USAGE
/*-------------
&type Usage: &r SELDATA <from_yr> <from_mt> <to_yr> <to_mt>
<data_info>
&type              <station_cover> <join_item> <out_cover> {YES|NO}
&return &inform
/*-------------
&routine EXIT
/*-------------
/* delete all temporary files:
&if [exists %pultmp% -info ] &then ; &s x = [delete %pultmp% -info]
&if [exists %xxxout%tmp -COVER] &then ; kill %xxxout%tmp all
&messages &on
&return
/*--------------
&routine BAILOUT
/*--------------
&severity &error &ignore
&call exit
&return &warning An error has occurred in SELDATA.AML
```

```
/* ----------------------------------------------------------------
/*     EXAMPLE OF APPLICATION
/*-----------------------------------------------------------------
/*      /* input
/* &s data = $HOME/iowa/data/gsflow.pat /* Info file of monthly flows
/* &s head = $HOME/iowa/data/crfhd    /* coverage of gauging stations
/* &s cname  = station_id             /* common item
/*     /* output
/* &s fout = mflow91                  /* output coverage
/*
/* &run seldata 1991 1 1991 12 %data% %head% %cname% %fout% NO
/*
/* &return                           /* return to Arc/Info prompt
/* ----------------------------------------------------------------
```

C7 FD4Y.AML

```
/* ---------------------------------------------------------------
/* fd4y.aml
/*         1) Creates ASCII files that contain data required
/*            for C program fd4y. ( fd4y calculates flow rate in
/*            all modeling units based on the flow recorded in
/*            USGS gauging stations).
/*         2) Stores the results obtained from fd4y program in
/*            PAT--polygon attribute table


&s prc = unprec.pat
&s flw = gsfl28.pat
&s out = unflow.pat
&s fy = 1960
&s fm = 1
&s ty = 1992
&s tm = 9

/*  input1 xxxgsin, ASCII file
/*  input2 xxxunin, ASCII file
/*  output xxxunout, ASCII file

tables

&DO yr = %ty% &to %fy% &by -1
  &DO mt = 12 &to 1 &by -1
    &s ab = [calc %yr% = %fy%] AND [calc %mt% lt %fm%]
    &s bb = [calc %yr% = %ty%] AND [calc %mt% gt %tm%]
    &s cc = %ab% OR %bb%
    &IF NOT %cc%  &THEN
      &do
        &s templ = mxxxx0y
        &s nitem = [subst %templ% xxxx %yr% ]
        &if %mt% lt 10 &then
            &sv nitem = [subst %nitem% y %mt% ]
          &else
            &sv nitem = [subst %nitem% 0y %mt% ]
        &type  processing item %nitem% ... [date -vfull]
/* ========================================================
select %flw%
unload xxxgsin station_id station_nx %nitem% DELIMITED INIT
select %prc%
unload xxxunin unit_id unit_nx gswsh area_km2 order p%nitem%
DELIMITED INIT
&sys fdy4 xxxgsin xxxunin xxxunout

&s i = 0
&do &until [exists xxxunout -file]
  &s i = %i% + 1
```

```
&end

define %nitem%.dat
unit_id,4,8,B
q%nitem%,4,12,F,4
~
add from xxxunout
select %nitem%.dat

&s x = [delete xxxunout -file]
&if %x% eq 0 &then
   &type file xxxunout has been deleted
&else
   &type ERROR: could not delete file xxxunout

&s x = [delete xxxgsin -file]
&if %x% eq 0 &then
   &type file xxxgsin has been deleted
&else
   &type ERROR: could not delete file xxxgsin

&s x = [delete xxxunin -file]
&if %x% eq 0 &then
   &type file xxxunin has been deleted
&else
   &type ERROR: could not delete file xxxunin


&sys arc joinitem %out% %nitem%.dat %out% unit_id order
kill %nitem%.dat

      &end
 &end
&end

q
q
&return
```

C8 SLOPE3.AML

```
/* -------------------------------------------------------------------
/*   Program: FSLOPE.AML
/*   Purpose: Calculates slope along the flow path. Calculates the
/*            difference between the elevation of the current cell
/*            and the elevation of the next cell on the flow pathand
/*            divides the result by the distance between these cells
/*            Runs only in GRID
/* -------------------------------------------------------------------
/*     Usage: &r FSLOPE <fdir> <dem> <fslp>
/* Arguments: (input)
/*            fdir = (grid) flow direction
/*            dem  = (grid) elevation
/*            (output)
/*            fslp = (grid) slope grid
/* Temporary: none
/* -------------------------------------------------------------------
/*     Notes: If flow direction is
/*            different than 1, 2, 4, 8, 16, 32, 64, and 128,
/*            the slope  0 is assumed.
/*            Vertical and horizontal units shuld be the same
/* -------------------------------------------------------------------
/*   History: Coded by Pawel Mizgalewicz  10/02/94 (slope3.aml)
/*            edited 16/06/95, 21/06/95, 02/24/96
/*            ( version with DOCELL -> slope3.aml)
/*            ( version with CONdition -> slope2.aml )
/*  ==================================================================

&args fdir dem fslp
&severity &error &routine bailout

&if [SHOW PROGRAM] <> GRID &then ; &return This only runs in GRID.
&if [null %fslp%] &then
 &do
  &call usage
  &return
 &end

&messages &off
/* GRID settings:
&s xxcell = [show setcell]
&s xxwindow = [show setwindow]
setcell minof
setwindow maxof
/* -------------------------------------------------------------------
&type calculating slope ... [date -vfull]
/* -------------------------------------------------------------------
&describe %dem%
&sv diag = 1.414213562 * %grd$dx%
DOCELL
  if (%fdir% == 1)
```

```
    %fslp% = ( %dem% - %dem%(1,0) ) / %grd$dx%
  else
    if (%fdir% == 2)
      %fslp% = ( %dem% - %dem%(1,1) ) / %diag%
    else
      if (%fdir% == 4)
        %fslp% = ( %dem% - %dem%(0,1) ) / %grd$dx%
      else
        if (%fdir% == 8)
            %fslp% = ( %dem% - %dem%(-1,1) ) /  %diag%
        else
          if (%fdir% == 16)
              %fslp% = ( %dem% - %dem%(-1,0) ) / %grd$dx%
          else
            if (%fdir% == 32)
               %fslp% = ( %dem% - %dem%(-1,-1) ) / %diag%
            else
              if (%fdir% == 64)
                 %fslp% = ( %dem% - %dem%(0,-1) ) / %grd$dx%
              else
                if (%fdir% == 128)
                   %fslp% = ( %dem% - %dem%(1,-1) ) / %diag%
                else
                   %fslp% = 0
END
&type end of fslope [date -vfull]
/* restore GRID settings:
setcell %xxcell%
setwindow %xxwindow%
&messages &on
&return
/*-------------
&routine USAGE
/*-------------
&type Usage: &r FSLOPE <fdir_grid> <elevation_grid> <slope_grid>
&return &inform
/*-------------
&routine EXIT
/*-------------
/* on error delete slope grid:
&if [exists %fslp% -GRID] &then ; kill %fslp% all
/* restore GRID settings:
setcell %xxcell%
setwindow %xxwindow%
&return
/*--------------
&routine BAILOUT
/*--------------
&severity &error &ignore
&call exit
&return &warning An error has occurred in FSLOPE.AML
/* -------------------------------------------------------------------
/*     EXAMPLE OF APPLICATION slope of stream reaches
/* -------------------------------------------------------------------
/* grid                                /* start GRID
/*      /* input
/* &s fdir = $HOME/iowa/data/crfdr
/*                              /* crfdr = flowdirection(elev_grid)
/* &s dem  = $HOME/iowa/data/crdem     /* elevation grid
/* &s fstr = crstr
/*     /* output
/* &s fslp = crslp                     /* slope grid
/* &s strs = strslp                    /* stream slope
/*
/* /* create slope grid
/* &run fslope %fdir% %dem% %fslp%
/*
/* /* assign unique number to each reach
/* xxxlsl = streamlink(%fstr%)
/*
/* /* calculate average slope
/* %strs% = zonalmean ( xxxlsl , %fslp% )
/*
/* kill xxxlsl all                     /* delete temporary grid
/* kill %fslp% all                     /* delete slope grid
/* q                                   /* quit from GRID
/*
/* &return                             /* return from example AML
/* ----------------------------------------------------------------
```

C9 DEMALB.AML

```
/* DEMALB.AML - converts a grid into Albers Equal Area coordinates
&args input output csize

%output% = project (%input% , #, NEAREST, %csize% )
OUTPUT
Projection    ALBERS
Zunits        NO
Units         METERS
Datum         NAD83
Spheroid      GRS1980
Xshift        0.0000000000
Yshift        0.0000000000
Parameters
 29 30  0.000 /* 1st standard parallel
 45 30  0.000 /* 2nd standard parallel
-96  0  0.000 /* central meridian
 23  0  0.000 /* latitude of projection's origin
0.00000 /* false easting (meters)
0.00000 /* false northing (meters)
END
&return
```