Part 618 – Soil Properties and Qualities

Subpart B – Exhibits

618.103 NASIS Calculation for Estimating Particle Size

Definition.—This calculation computes representative values for the sand fractions, total sand, and total silt. The following rules apply:

1) The results will be blank if needed data are not entered. Total clay and texture are always required, and particle size class is required for textures CL, L, SCL, SICL, and SIL.
2) When a horizon has multiple textures, the one marked rv is used or the first texture is used if there is no rv. No results are calculated for stratified textures at this time.
3) If total sand (rv) has been entered, the sand fractions will be adjusted so their sum equals the specified total. If you want to calculate a new sand total, you must erase the old one before running the calculation.

Inputs.—This calculation requires that the following data must be populated:

- texture
- clay total separate (l, rv, h)
- taxonomic particle-size class

Calculation.

ASSIGN texcl IF ISNULL(texcl) OR stratextsflag==1 THEN "null" ELSE CODENAME(texcl).

DEFINE sandclass
   IF (texcl=="sl" or texcl=="cosl" or texcl=="fsl" or texcl=="vfsl") THEN
      IF ISNULL(sandtotal_r) THEN 1 ELSE
      IF sandtotal_r > 60 THEN 1 ELSE
      IF sandtotal_r >= 53 THEN 2 ELSE 3
   ELSE IF (texcl=="cl" or texcl=="l" or texcl=="scl" or texcl=="sicl" or texcl=="sil") THEN family_sandclass
   ELSE 0.

DEFINE paramid_by_tex LOOKUP(1, texcl==texture and (sandcode==0 or sandcode==sandclass), paramid).
DEFINE claypct_by_tex LOOKUP(1, texcl==texture and (sandcode==0 or sandcode==sandclass), claypct).
DEFINE claydiff_by_tex ABS(claypct_by_tex - claytotal_r).
DEFINE closest_clay ARRAYMIN(claydiff_by_tex).
DEFINE select_row ARRAYMIN(LOOKUP(closest_clay, claydiff_by_tex, paramid_by_tex)).

# Get the equation number and coefficients from the selected parameter row.

DEFINE eqn lookup(select_row, paramid, equation).
DEFINE p1 lookup(select_row, paramid, param1).
DEFINE p2 lookup(select_row, paramid, param2).
DEFINE p3 lookup(select_row, paramid, param3).
DEFINE p4  lookup(select_row, paramid, param4).
DEFINE p5  lookup(select_row, paramid, param5).

# Compute all the distributions. We compute all 5 equations first then
# pick the right result, because this language doesn't have conditional
# execution.
# Start by computing some things that are used more than once.

DEFINE diamclay LOOKUP("clay", psclass, psdiam).  # Upper clay diameter.
DEFINE cr2    POW(2, 1/3).              # Cube root of 2.
DEFINE crdiam POW(psdiam, 1/3).         # Cube root of psclass diameter.
DEFINE crdiamclay POW(diamclay, 1/3).  # Cube root of clay diam.
DEFINE sqrdiam SQRT(2).                 # Square root of 2.
DEFINE sqrdiam SQRT(psdiam).           # Square root of psclass diameter.

DEFINE eq1tmpl POW(1 + p4*POW(cr2-crdiam, p3), p2).
DEFINE eq2tmp EXP(p3 * POW(cr2-crdiam, p2)).
DEFINE eq3tmp EXP(1/POW(p4*cr2,p3) - 1/POW(p4*crdiam,p3)).
DEFINE eq4tmp p3*(POW(cr2,2) - POW(cr2,2)) + p4*(psdiam-2) +
p5*(POW(psdiam,4) - POW(psdiam,4)).
DEFINE eq5tmp p3*(1/psdiam - 1/2) + p4*(sqrdiam - sqrd).

# Next adjust the parameters to make the clay come out the same as the input.

DEFINE eq1p1 (tmp * claytotal_r - 100) / (tmp - 1).
DEFINE eq2p1 (tmp * claytotal_r - 100) / (tmp - 1).
DEFINE eq3p2 (claytotal_r - 100 * tmp) / (1 - tmp).
DEFINE eq4p2 (claytotal_r - 100 - tmp) / (crdiamclay - cr2).
DEFINE eq5p2 (claytotal_r - 100 - tmp) / (diamclay - 2).

# Compute the five equations for all particle size classes.

DEFINE eq1  eq1p1 + (100-eq1p1) / eq1tmp.
DEFINE eq2  eq2p1 + (100-eq2p1) / eq2tmp.
DEFINE eq3  eq3p2 + (100-eq3p2) * eq3tmp.
DEFINE eq4  100 + eq4p2*(crdiam - cr2) + eq4tmp.
DEFINE eq5  100 + eq5p2*(psdiam - 2) + eq5tmp.

# Select the right equation. The variable psd will have 7 values, one
# for each particle size class. The value for each class is picked out
# of the array with a LOOKUP.

DEFINE psd IF eqn==1 THEN eq1 ELSE
            IF eqn==2 THEN eq2 ELSE
            IF eqn==3 THEN eq3 ELSE

(430-VI-NSSH, Amend. 103, November 2010)
IF eqn==4 THEN eq4 ELSE
IF eqn==5 THEN eq5 ELSE
eq5/0.               # sets psd to 7 nulls when texcl is null

# Pick out the cumulative percents then compute the individual fractions.
DEFINE clay LOOKUP("clay", psclass, psd).
DEFINE silt LOOKUP("silt", psclass, psd).
DEFINE vfs LOOKUP("vfs", psclass, psd).
DEFINE fs  LOOKUP("fs", psclass, psd).
DEFINE ms  LOOKUP("ms", psclass, psd).
DEFINE cs  LOOKUP("cos", psclass, psd).
DEFINE vcs LOOKUP("vcos", psclass, psd).
ASSIGN vcs  vcs - cs.
ASSIGN cs   cs - ms.
ASSIGN ms   ms - fs.
ASSIGN fs   fs - vfs.
ASSIGN vfs vfs - silt.
ASSIGN silt silt - clay.

DEFINE sand vfs + fs + ms + cs + vcs.

# Find an adjustment factor for the sand fractions.
# If total sand was given, adjust each sand fraction by the ratio needed to
# make the sum equal to the given total.
# If total sand was not given, verify that the sand and silt are within the
# texture class limits and if not adjust them by the appropriate ratio.
DEFINE sand_diff IF ISNULL (sandtotal_r) THEN
    IF (texcl=="cos" or texcl=="s" or texcl=="fs" or texcl=="vfs")
        and ((clay + silt) > (15 - .5*clay))
        THEN (clay + silt) - (15 - .5*clay) ELSE
        IF (texcl=="lcos" or texcl=="ls" or texcl=="lfs" or texcl=="lvfs")
            and (clay + silt) > (30 - clay)
            THEN (clay + silt) - (30 - clay) ELSE
            IF texcl=="sil" and silt < 50
                THEN silt - 50 ELSE
                IF texcl=="sicl" and sand > 20
                    THEN 20 - sand ELSE
                    IF texcl=="sc" and sand < 45
                        THEN 45 - sand ELSE
                        IF texcl=="sic" and silt < 40
                            THEN silt - 40 ELSE
                            0
                        ELSE
                            sandtotal_r - sand.
    DEFINE adj (sand + sand_diff) / sand.

# Adjust the sands and silt by the adjustment factor.
# Round to one decimal place before computing total sand to avoid roundoff error.
ASSIGN vfs  ROUND(vfs * adj, 1).
ASSIGN fs   ROUND(fs  * adj, 1).
ASSIGN ms   ROUND(ms  * adj, 1).

(430-VI-NSSH, Amend. 103, November 2010)
ASSIGN cs  ROUND(cs * adj, 1).
ASSIGN vcs  ROUND(vcs * adj, 1).
ASSIGN sand  vfs + fs + ms + cs + vcs.

# The rounding may result in a sum that does not equal the target sandtotal,
# so another adjustment has to be made.
# This time, apply it to the first non-zero fraction.

ASSIGN adj  IF NOT ISNULL(sandtotal_r) THEN sandtotal_r - sand ELSE 0.

ASSIGN vcs  IF vfs==0 AND fs==0 AND ms==0 AND cs==0 THEN vcs + adj ELSE vcs.
ASSIGN cs  IF vfs==0 AND fs==0 AND ms==0 AND cs>0 THEN cs + adj ELSE cs.
ASSIGN ms  IF vfs==0 AND fs==0 AND ms>0 THEN ms + adj ELSE ms.
ASSIGN fs  IF vfs==0 AND fs>0 THEN fs + adj ELSE fs.
ASSIGN vfs  IF vfs > 0 THEN vfs + adj ELSE vfs.
ASSIGN sand  vfs + fs + ms + cs + vcs.

ASSIGN silt  100 - sand - clay.

# When vcos is < 0.
ASSIGN cs  IF vcs < 0 AND vfs+fs+ms+cs > sand THEN cs - ((vfs+fs+ms+cs) - sand) ELSE cs.
ASSIGN vcs  IF vcs < 0 THEN 0 ELSE vcs.

# Store the results as RV values for the horizon.

SET sandtotal_r from sand,
   sandvc_r from vcs,
   sandco_r from cs,
   sandmed_r from ms,
   sandfine_r from fs,
   sandvf_r from vfs,
   silttotal_r from silt.

(430-VI-NSSH, Amend. 103, November 2010)