1. Fineroot turnoverate modeling

Fineroot turnoverrates were based on Kleja et al. (2008) and to Leppälammi-Kujansuu et al. (2014). It was assumed that the turnoverrate of fineroots is 85% in Southern Finland in areas where temperature sum during the growing season (with 5 C^{o} treshold) is more than 1200. For Northern Finland we used turnoverate of 50% when temperature sum was less than 700. For sites where temperature sums were between 700 and 1200 we interpolated the turnoverrate with a simple linear regression,

$y = b_0 + b_1 \times x$, where

y is the turnoverrate and b_0 equals 0.0007, b_1 equals 0.001 and x is degree days that varies between 700 and 1200 degrees.

2. Estimation of understorey biomass

Here we provide parameter estimates for understorey models (Table S1) and figures for model fits (Figs. S1 & S2).

Table S1. Parameter estimates for understorey models by species groups and regions. Where β_0 is the intercept and β_1 is a slope of the fixed part of the model and bc is the bias correction. Var b_0, Cov $b_o b_1$ and Var b_1 originate from variance-covariance matrix of the fixed part of the model. Var pop lists unexplained variance and Var plt lists unexplained variance after plot as a random effect.

Group	Com.	Reg.	β_0	β_1	bc	Varβ_0	Covβ_o β_1	Varβ_1	Var Pop	Var Plt
Dwarf shrub	Ab	NF	3.62	0.948	0.135	0.115	-0.028	0.008	0.285	0.136
Dwarf shrub	Ab	SF	2.653	1.107	0.186	0.081	-0.015	0.005	0.45	0.16
Grass	Ab	-	2.75	0.918	0.292	0.129	-0.03	0.01	0.688	0.273
Herb	Ab	-	1.116	1.08	0.445	0.074	-0.018	0.008	0.832	0.572
Bryophytes	Ab	-	3.13	0.795	0.281	0.301	-0.072	0.018	0.633	0.306
Lichen	Ab	-	3.69	0.894	0.109	0.047	-0.012	0.005	0.27	0.175
Shrub	Bl	NF	6.278	0.45	0.161	0.086	-0.02	0.006	0.315	0.241
Shrub	Bl	SF	3.73	0.831	0.458	0.15	-0.035	0.011	1.06	0.867
Grass	Bl	NF	3.368	0.589	0.639	0.169	-0.039	0.017	1.423	0.859
Herb	Bl	SF	1.877	0.906	0.709	0.27	-0.067	0.03	1.195	0.898

Com. = compartment: Ab = aboveground, Bl = belowground
Reg. = region: NF = northern Finland, SF = southern Finland

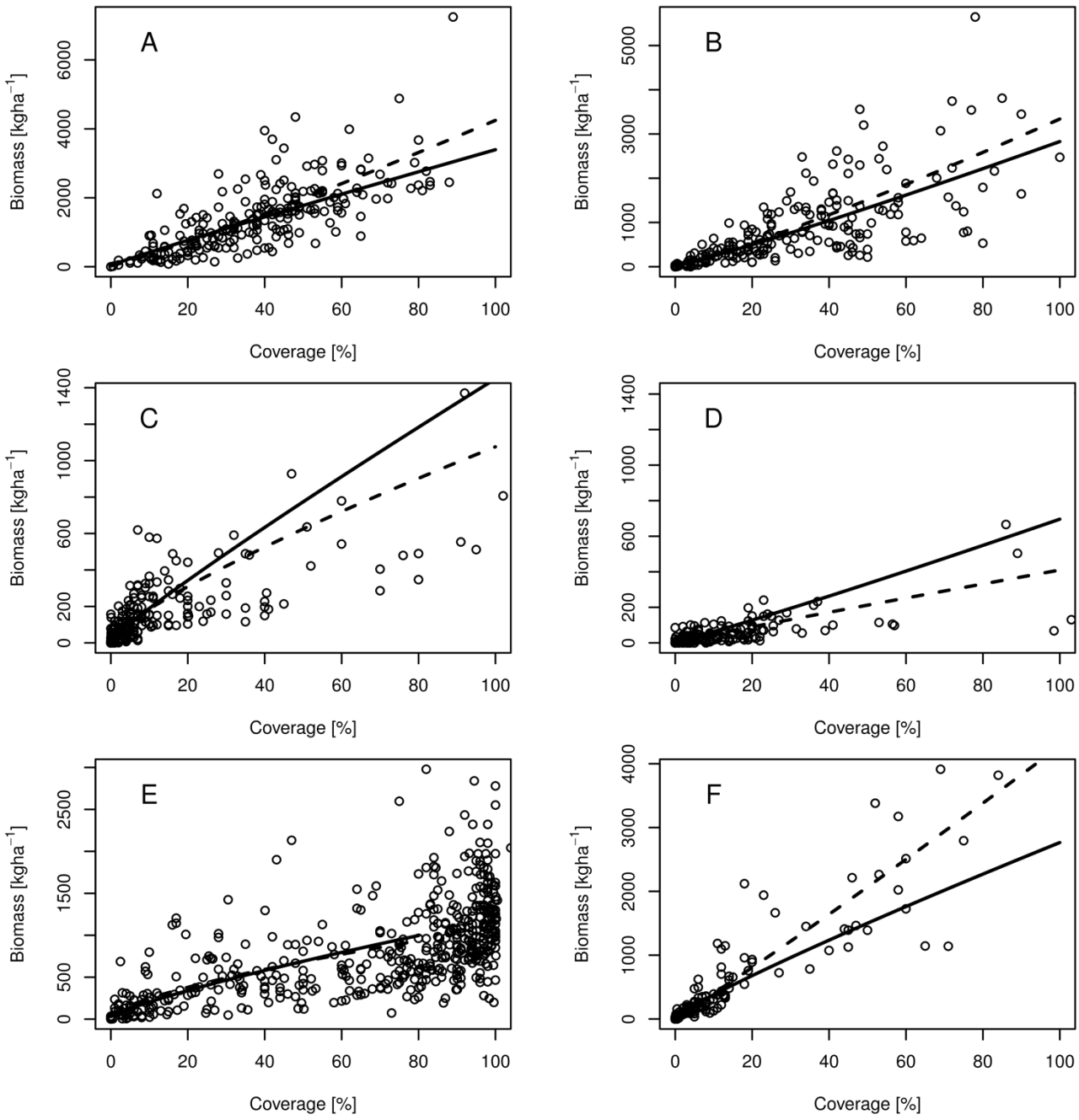

Figure S1: Models and data points for aboveground understorey biomass (A = Dwarf shrubs in northern Finland, B = Dwarf shrubs in southern Finland, C = Grasses, D = Herbs, E = Mosses and F = Lichen). Solid line based on modelling that takes into account site fertility distribution for Finland by weighting, while dashed line is the estimate without weights.

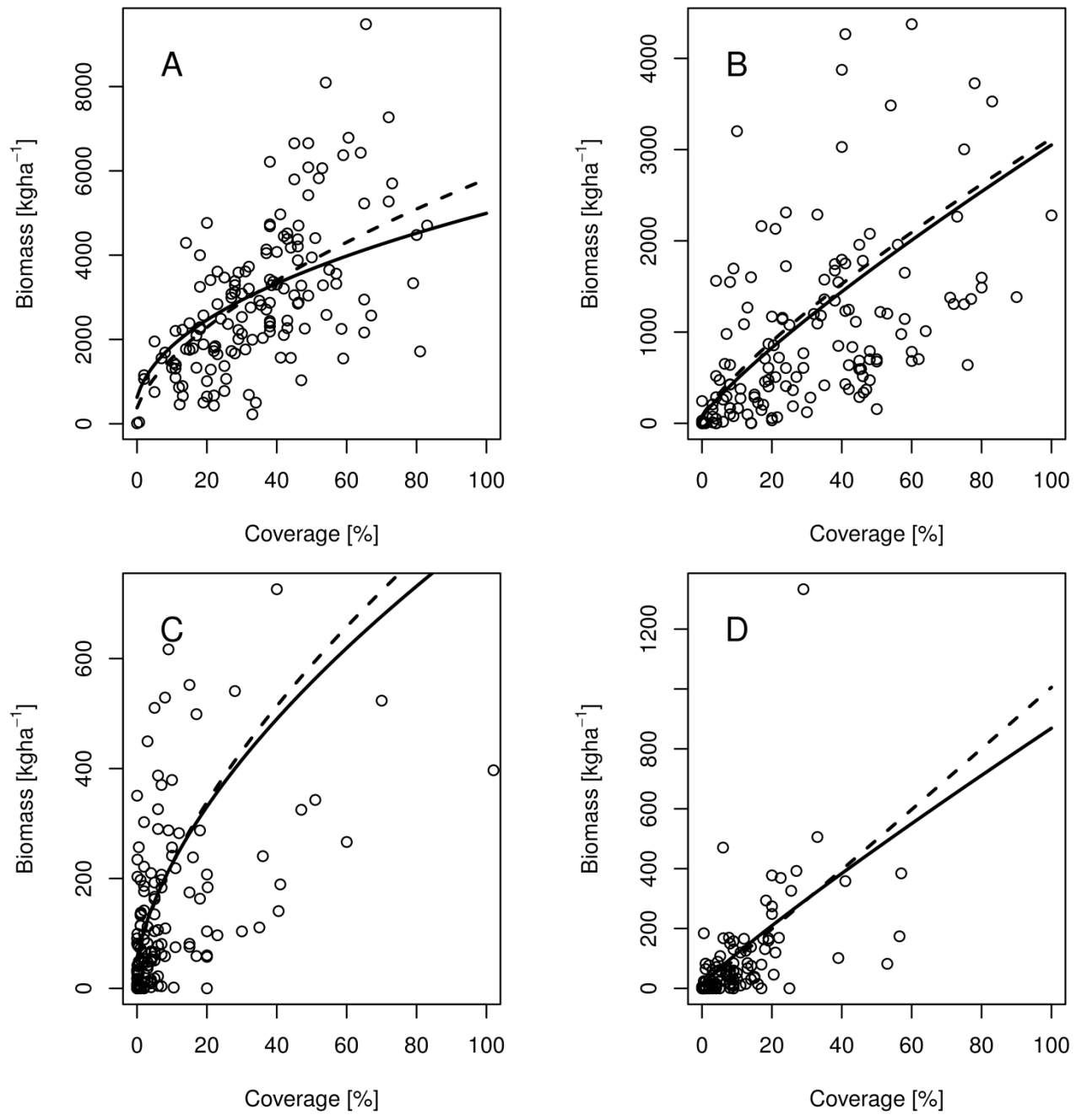

Figure S2. Models and data points for belowground understorey biomass (A = Dwarf shrubs in northern Finland, B = Dwarf shrubs in southern Finland, C = Grasses and D = Herbs). Solid line based on modelling that takes into account site fertility distribution for Finland by weighting, while dashed line is the estimate without weights.

R code for variograms and cross-variograms of vegetation group coverages and resulting figure with semivariance as a function of distance (Fig. S3).

```
library(gstat)

library(sp)

### peite34 object includes data for vegetation coverages and plot locations across Finland

coordinates(peite34) = c('x','y')

####################

g.r <- gstat(NULL, "dwarf shrubs", r1~x+y,data = peite34,  model = v.fitr1, nmax=200)

g.r <- gstat(g.r, "grass&herb", r2~x+y,data = peite34,  model = v.fitr2, nmax=200)

g.r <- gstat(g.r, "bryophyte", r3~x+y,data = peite34,  model = v.fitr3, nmax=200)

g.r <- gstat(g.r, "lichen", r4~x,data = peite34,  model = v.fitr4, nmax=200)

vm <- variogram(g.r)

vm.fit <- fit.lmc(vm,g.r,vgm(1200,"Sph",100,300))

plot(vm, vm.fit)

## thereafter predictions of the understorey coverage for each location by vegetation groups

cok.maps <- predict(vm.fit, nfi10loc)
```

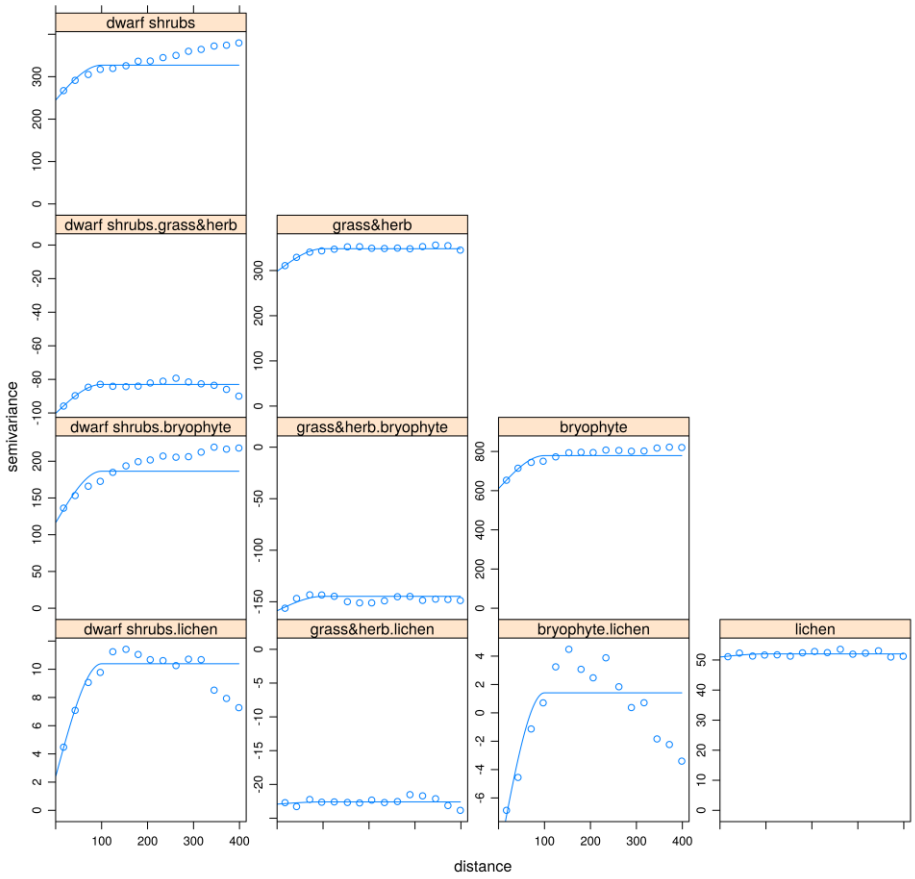

Figure S3. Variograms and cross-variograms for different functional types of understorey vegetation.

3. Nitrogen content of the litterfall

Table S2. Nitrogen content of litterfall as ratios from mass, based on Komarov et al. (2007).

	Foliage	Branches	Stem wood	Fine roots	Coarse roots	Total
Scots pine	0.003	0.004	0.0014	0.0047	0.0024	-
Norway spruce	0.0045	0.0035	0.002	0.0035	0.003	-
Deciduous	0.007	0.004	0.0015	0.005	0.0045	-
Understorey[1]	-	-	-	-	-	0.006

4. Soil carbon model Yasso07

Parameters for the Yasso07 have been estimated with Markov Chain Monte Carlo (MCMC) methods where litter bag, wood decomposition and soil carbon stock data were used to calibrate model parameters. Maximum posterior parameters estimates used in this study were based on works by Rantakari et al. (2012) and for earlier version of the global Yasso07 parameterisation that is a close

[1] The nitrogen content of the understorey litterfall was estimated to be 0.6% from dry mass.

variant of the Tuomi et al. (2011) publication (Table S2). Model have been described more in detail in papers where individual components have been reported (Tuomi et al. 2009, Tuomi et al. 2011, Tuomi et al. 2008)

Fortran code of the Yasso07 model is available here:

http://code.google.com/p/yasso07ui/

Table S2. Yasso07 maximum *a posterior* (MAP) point estimates Scandinavian (Rantakari et al. 2012) and global parameterization (Tuomi et al. 2011).

Parameter	Scandinavia	Global	Unit	Meaning
aA	0.52	0.72	a^{-1}	decomposition rate of A
aW	3.55	5.9	a^{-1}	decomposition rate of W
aE	0.35	0.28	a^{-1}	decomposition rate of E
aN	0.27	0.031	a^{-1}	decomposition rate of N
p1	0.04	0.48	.	mass flow from W to A
p2	0.03	0.01	.	mass flow from E to A
p3	0.98	0.83	.	mass flow from N to A
p4	0.64	0.99	.	mass flow from A to W
p5	0.31	0.00	.	mass flow from E to W
p6	0.019	0.01	.	mass flow from N to W
p7	0.023	0.00	.	mass flow from A to E
p8	0.01	0.00	.	mass flow from W to E
p9	0.001	0.02	.	mass flow from N to E
p10	0.34	0.00	.	mass flow from A to N
p11	0.042	0.015	.	mass flow from W to N
p12	0.09	0.95	.	mass flow from E to N
b1	0.09	0.95	C^{-1}	temperature dependence parameter
b2	-0.0023	-1.4	10^{-}	temperature dependence parameter
y	-2.94	-1.21	m^{-1}	precipitation dependence parameter
pH	0.15	4.5	10^{-3}	mass flow from A,W,E,N to humus
aH	-0.24	-1.6	10^{-3}	humus decomposition coefficient
roo1	-0.539	-1.71	cm^{-1}	size dependence parameter
roo2	1.186	0.86	cm^{-2}	size dependence parameter
r	-0.264	-0.306	.	size dependence parameter

5. Soil carbon model ROMULv

ROMUL decomposition model describes the flux of soil organic matter (SOM) through the soil

decomposition process, divided into separate parallel paths of matter based on the different origins of the litter (Chertov and Komarov 1997, Chertov et al. 2001). All fluxes have essentially the same pattern of flux. The litter entering the decomposition process is stored in a store of undecomposed litter (H). This then decomposes into a mixture of partly decomposed (humified) SOM (F). The SOM in the F fraction is decomposed by different types of organisms (fungi, bacteria and earthworms) and end up in storage of semi-stable humus (H). These fluxes (for litter of different origins) are for SOM, but the model also describes a parallel decomposition process for nitrogen in soil, following the same pattern of storages and fluxes.

Litter → L → F → H

In each transition from one storage to another, the rate of decomposition depends on the nitrogen content of the storage (calculated from the ratio of the SOM storage and the corresponding N storage) and the ash content of the litter (fixed percentage for each litter source). These transfer rates are adjusted with coefficient functions depending on soil temperature and soil water content. During each transition, a fraction of the pool is mineralized – SOM released as CO_2 and N released into a pool of mineralized N available to plants, while most of the matter moves to the next pool in the decomposition process. The litter from different sources follow different routes, until they all end up in a common pool of semi-stable humus (H). Finally, also the matter in the H pool slowly decomposes. In each transition from one pool to another, the rate of SOM and N decomposition is the same, except for the transition from F pool to H; here the C:N ratio of the transition is characteristic for all decomposer classes (fungi, bacteria and earthworms), and in this phase the N content of the soil is enriched.

The litter from aboveground (leaves, shoots and trunks) fall onto the forest floor, whereas the root litter (fine roots and coarse roots) decomposes in the mineral soil layer. For the two layers, the decomposition rates based on SOM properties are mostly similar, but the environmental conditions (temperature and soil moisture) may differ for the two layers, resulting in different decomposition rates. The semi-stable storage H is common for all fluxes and resides in the mineral soil layer.

The original ROMUL model uses decomposition rate functions that use gravimetric measures of soil

water as input. As these figures are somewhat tedious to calculate and would require detailed information of the soil characteristics, such as soil density and precise thickness of the soil layers, we used instead a modification presented by Linkosalo et al. (2013) named here as ROMULv. In their paper they produced decomposition rate functions that are based on volumetric soil water measures, and a volumetric soil water model to predict the variation in the soil water content, driven by environmental measures. The volumetric soil water model was easier to apply for multiple simulation points, as it only needs the total potential soil water storage (i.e. difference between field capacity and wilting point) to characterize the soil for the simulations of relative soil water content.

The description of the original ROMUL model is available from here:

http://ecomodelling.ru/

For the source code used in this study, see the end of this document.

References

Chertov, O. G. and Komarov, A. S.: SOMM: A model of soil organic matter dynamics, Ecol. Modell., 94, 177-189, doi:10.1016/S0304-3800(96)00017-8, 1997.

Chertov, O. G., Komarov, A. S., Nadporozhskaya, M., Bykhovets, S. S. and Zudin, S. L.: ROMUL — a model of forest soil organic matter dynamics as a substantial tool for forest ecosystem modeling, Ecol. Model., 138, 289-308, doi:10.1016/S0304-3800(00)00409-9, 2001.

Kleja, D. B., Svensson, M., Majdi, H., Jansson, P., Langvall, O., Bergkvist, B., Johansson, M., Weslien, P., Truusb, L. and Lindroth, A.: Pools and fluxes of carbon in three Norway spruce ecosystems along a climatic gradient in Sweden, Biogeochemistry, 89, 7-25, doi:10.1007/s10533-007-9136-9, 2008.

Komarov, A. S., Chertov, O. G., Nadporozhskaya, M. A., Mikhailov, A. V., Bykhovets, S. S., Zudin, S. L., Priputina, I. V., Larionova, A. A., Grabarnik PY, Z. E., Zudina, E. V. and Shanin, V. N.: Modelirovanie dinamiki organicheskogo veshchestva v lesnykh ekosistemakh [Modelling Organic Matter Dynamics in Forest Ecosystems], Nauka, Russia, 2007.

Leppälammi-Kujansuu, J., Aro, L., Salemaa, M., Hansson, K., Kleja, D. B. and Helmisaari, H.: Fine root longevity and carbon input into soil from below- and aboveground litter in climatically contrasting forests, For. Ecol. Manage., 326, 79-90, doi:10.1016/j.foreco.2014.03.039, 2014.

Linkosalo, T., Kolari, P. and Pumpanen, J.: New decomposition rate functions based on volumetric soil water content for the ROMUL soil organic matter dynamics model, Ecol. Model., 263, 109-118, doi:10.1016/j.ecolmodel.2013.04.026, 2013.

Rantakari, M., Lehtonen, A., Linkosalo, T., Tuomi, M., Tamminen, P., Heikkinen, J., Liski, J., Mäkipää, R., Ilvesniemi, H. and Sievänen, R.: The Yasso07 soil carbon model – Testing against repeated soil carbon inventory, For. Ecol. Manage., 286, 137-147, doi:10.1016/j.foreco.2012.08.041, 2012.

Tuomi, M., Rasinmäki, J., Repo, A., Vanhala, P. and Liski, J.: Soil carbon model Yasso07 graphical user interface, Environ. Model Softw., 26, 1358-1362, doi:10.1016/j.envsoft.2011.05.009, 2011.

Tuomi, M., Thum, T., Järvinen, H., Fronzek, S., Berg, B., Harmon, M., Trofymow, J. A., Sevanto, S. and Liski, J.: Leaf litter decomposition--Estimates of global variability based on Yasso07 model, Ecol. Model., 220, 3362-3371, 2009.

Tuomi, M., Vanhala, P., Karhu, K., Fritze, H. and Liski, J.: Heterotrophic soil respiration--Comparison of different models describing its temperature dependence, Ecol. Model., 211, 182-190, 2008.

Source code of the ROMULv model used here

```
Main program loop (meta-code):

Initialize ROMUL (read initial values from file)

Loop_over_data {

  read meteorological data
  calculate soil temperature

  calculate evapotranspiration
  calculate soil water status

  calculate decomposition and update soil storages
}

write ROMUL results
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c       This implementation of the ROMUL model is based on
c       Chertov et al. 2001, with "new functions" from
c       Komarov et at. 2008.
c
c       Tapio Linkosalo (METLA) & Annikki Mäkelä (UH)
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

```
        subroutine ROMUL(litter,litter_N,T,SW,Navail,
     +                site, year, init)

        implicit none

c litterfall by cohort, soil temperature and water content for organic (==1)  and mineral soil layer
(==2)
c init-parameter for initializing the model (1), normal simulation (0) or output the results (2).

       double precision litter(11), litter_N(11)
       real T(2), SW(2)

c mass storages are save-variables, values persist from one call to another
c First dimension of pools is for the SOM cohort, the second for spatial locations
c (except only one pool per geographic location for humus)
       double precision Lpool(11,4000), Fpool(11,4000)
       double precision LNpool(11,4000), FNpool(11,4000)
       double precision Hpool(4000), HNpool(4000)
       save Lpool, Fpool, Hpool, LNpool, FNpool, HNpool

       double precision Navail(4000)
        real time, Nup, fnf, gnf, fnwb, Ndemand, qctot
        integer site, year, init

       integer local_year
       static local_year

c declare decomposition coefficient functions
       real f_1, f_2, f_3, f_4, f_5, f_6
       real g_1, g_2, g_3, g_4, g_5, g_6
c
c * N pools etc
c
c  litter in 11 separate pools
c  all have L and F fractions, one H fraction
c  all fractions have SOM and N
c  cohorts are order thus:
c     1   foliage
c     2   branches
c     3   stems
c     4   fine roots
c     5   coarse roots
c     11  ground vegetation
c  Note! Cohorts 6 to 10 are for felling residues, but in this version
c  they are calculated in corresponding litter cohorts (6->1 etc.)

c parameters - following ROMUL
c
c       kL             specific rate of leaching (yr-1)     1.0
```

```
c       kDL(i)  specific rate of litter decomposition (yr-1)       0.3
c       kDF(i)  specific rate of SOM decomposition (yr-1)    0.3
c       kTL(i)  specific rate of transfer from litter to SOM  (yr-1)
c       kTF(i)  specific rate of transfer from SOM to humus, comp. 1  (yr-1)       0.3
c       kTF2(i) specific rate of transfer from SOM to humus, comp. 2  (yr-1)       0.3
c       kDH             specific rate of humus decomposition (yr-1) 1/6000
c       kUG             maximum specific N uptake rate of ground vegetation (yr-1) 100
c       DeposN  N deposition (free input) (kg yr-1)  1
c
c     ash_cont(i) ash content of cohorts
c     fnf     foliar N concentration in live foliage
c       gnf             retention of foliar N when shedding foliage 0.45
c
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c Declare internal variables

      double precision DeposN, LeachN
       double precision d_Lpool(11),d_Fpool(11),d_HumusPool
      double precision d_LNpool(11),d_FNpool(11),d_HumusNPool
       double precision d_Navail
      double precision fn(11)
       real kL, kM, DM
      double precision kDL(11), kDH
      double precision kDF(11)
      double precision kTL(11)
      double precision kTF(11),kTF2(11)
      real LKP_decomp
      double precision K_1S_rma, K_2S_rma,K_3S_rma, k_1L_rma,K_2L_rma
      double precision k_3L_rma, k_4_rma,k_5_rma,k_6_rma
      double precision ML, MF(11), MFres(11), MH, gamma, GVN1, GVN2
       double precision N_release, FH_FluxB, FH_fluxL, FHN_FluxB, FHN_fluxL
      double precision MF_flux, MF_min
      double precision H_miner, H_C, C_C, DeltaB, DeltaL, dL
      double precision x1, x2, ash_cont(11)
      integer inttime, i, j, k, inttimetot
      double precision Lpoolsum(5),Fpoolsum(5),LNpoolsum(5),FNpoolsum(5)
      double precision littersum(5), Fpooltot(2), FNpooltot(2), H_CN
      double precision Lpooltot(2), LNpooltot(2)
      double precision NFconc(2), NLconc(2)
       double precision step


c assign parameter values
       data kL
     1 /0.12/

      data (ash_cont(i) , i=1,11)
     1 /0.02,  0.02,  0.01,  0.02,  0.02, 0.02, 0.02, 0.01,
```

```
     1 0.02, 0.01, 0.04/

      data fnf, gnf  /0.02, 0.45/

      data DeposN, kM, DM /2., 2., 0.3/
      data ML, gamma, C_C / 0.1, 0.8, 0.5/
      data deltaB, deltaL /24., 12.8/


      dL = 1




c read inputs when coming to this subroutine for the first time (init = 1)
      if (init.eq.1) then
          open(unit=25,file='soil.dat',status = 'old', err = 999)
         do 100 i = 1, site
          read(25,*) j, (Lpool(j,i),j=1,5),Lpool(11,i),
     +                     (Fpool(j,i),j=1,5),Fpool(11,i),
     +                        Hpool(i),
     +            (LNpool(j,i),j=1,5),LNpool(11,i),
     +            (FNpool(j,i),j=1,5),FNpool(11,i),
     +            HNpool(i)

100      continue
          close(25)
         goto 998
999      write(*,*) "failed to open soil.dat"


         local_year = 0

998         return
      endif


c write storages to file when coming to this subroutine for the last time (init = 2)
      if (init.eq.2) then
          open(unit=25,file='soil.dat',status = 'old')

         write(*,*) "writing soil.dat..."
         do 101 i = 1, site
              write(25,997) i,(Lpool(j,i),j=1,5),Lpool(11,i),
     +                        (Fpool(j,i),j=1,5),Fpool(11,i),
     +         Hpool(i),
     +        (LNpool(j,i),j=1,5),LNpool(11,i),
     +        (FNpool(j,i),j=1,5),FNpool(11,i),
     +            HNpool(i)
997   format(i5,12f12.2,f14.2,12f12.5,f14.3)
```

```
101      continue
          close(25)
         return
      endif


c Daily calculation starts here!

         do 65 i = 1, 11
             fn(i) = litter_N(i)/litter(i)
65       continue

c "step" is for substeN of differential calculation, with daily weather data
c no substeN needed.
            step = 1

c compute specific rate parameters as functions of ash content and N content
c compute total N in litter

c first the N concentration for the F-pools:
c index 1 = organic layer, and 2 = mineral soil

      Lpooltot(1) = Lpool(1,site) + Lpool(2,site) + Lpool(3,site) +
     +               Lpool(11,site)
      LNpooltot(1) = LNpool(1,site) + LNpool(2,site) + LNpool(3,site) +
     +               LNpool(11,site)
      Fpooltot(1) = Fpool(1,site) + Fpool(2,site) + Fpool(3,site) +
     +               Fpool(11,site)
      FNpooltot(1) = FNpool(1,site) + FNpool(2,site) + FNpool(3,site) +
     +               FNpool(11,site)
      Fpooltot(2) = Fpool(4,site) + Fpool(5,site)
      FNpooltot(2) = FNpool(4,site) + FNpool(5,site)


      NFconc(1) = FNpooltot(1) / Fpooltot(1)
      NFconc(2) = FNpooltot(2) / Fpooltot(2)
      NLconc(1) = LNpooltot(1) / Lpooltot(1)
      NLconc(2) = LNpooltot(2) / Lpooltot(2)


      do 25 i = 1,5

          if(i.ge.4) then
              kDL(i) = k_1S_rma(ash_cont(i), fn(i)) *
     *            f_1(T(2)) * LKP_decomp(SW(2), 0.55)
             kDF(i) = k_2S_rma(ash_cont(i), NFconc(2)) *
     *            f_2(T(2)) * LKP_decomp(SW(2), 0.55)
              kTL(i) = k_3S_rma(ash_cont(i), fn(i)) *
     *            f_3(T(2)) *LKP_decomp(SW(2), 0.55)
          else
```

```
               kDL(i) = k_1L_rma(ash_cont(i), fn(i)) *
     *            f_1(T(1)) * LKP_decomp(SW(1), 0.7)
               kDF(i) = k_2L_rma(ash_cont(i), NFconc(1)) *
     *            f_2(T(1)) * LKP_decomp(SW(1), 0.7)
               kTL(i) = k_3L_rma(ash_cont(i), fn(i)) *
     *            f_3(T(1)) * LKP_decomp(SW(1), 0.7)
          endif

          kTF(i) = k_4_rma(ash_cont(i), NFconc(2)) *
     *            f_4(T(2)) * LKP_decomp(SW(2), 0.55)
          kTF2(i) = k_5_rma(ash_cont(i), NFconc(2)) *
     *            f_5(T(2)) * LKP_decomp(SW(2), 0.55)

25    continue

          kDL(11) = k_1L_rma(ash_cont(11), fn(11)) *
     *            f_1(T(1)) * LKP_decomp(SW(1), 0.7)
          kDF(11) = k_2L_rma(ash_cont(11), NFconc(1)) *
     *            f_2(T(1)) * LKP_decomp(SW(1), 0.7)
          kTL(11) = k_3L_rma(ash_cont(11), fn(11)) *
     *            f_3(T(1)) * LKP_decomp(SW(1), 0.7)

          kTF(11) = k_4_rma(ash_cont(11), NFconc(2)) *
     *            f_4(T(1)) * LKP_decomp(SW(1), 0.55)
          kTF2(11) = k_5_rma(ash_cont(11), NFconc(1)) *
     *            f_5(T(1)) * LKP_decomp(SW(1), 0.55)


       kDH = k_6_rma() * f_6(T(2)) * LKP_decomp(SW(2), 0.55)




c       update N pools

        Nup = 0.

c       do 100 inttime = 1, inttimetot



       LeachN = kL * Navail(site)
       LeachN = max(LeachN, 0.)


c C pools: derivatives

       FH_fluxB = 0.
       FH_fluxL = 0.
       FHN_fluxB = 0.
```

```
      FHN_fluxL = 0.

      do 10 i = 1,5

          d_Lpool(i) = litter(i) + litter(i+5) -
     -         (kDL(i) + kTL(i)) * Lpool(i,site)

          d_Fpool(i) = kTL(i) * Lpool(i,site)
     1                 -(kDF(i) + kTF(i) + kTF2(i)) * Fpool(i,site)

          FHN_fluxB = FHN_fluxB + kTF(i) * FNpool(i,site)
          FHN_fluxL = FHN_fluxL + kTF2(i) * FNpool(i,site)
          FH_fluxB = FH_fluxB + kTF(i) * Fpool(i,site)
          FH_fluxL = FH_fluxL + kTF2(i) * Fpool(i,site)

10    continue



          d_Lpool(11) = litter(11)  -
     -         (kDL(11) + kTL(11)) * Lpool(11,site)

          d_Fpool(11) = kTL(11) * Lpool(11,site)
     1                 -(kDF(11) + kTF(11) + kTF2(11)) * Fpool(11,site)

          FHN_fluxB = FHN_fluxB + kTF(11) * FNpool(11,site)
          FHN_fluxL = FHN_fluxL + kTF2(11) * FNpool(11,site)
          FH_fluxB = FH_fluxB + kTF(11) * Fpool(11,site)
          FH_fluxL = FH_fluxL + kTF2(11) * Fpool(11,site)



           d_HumusPool = DeltaB * FHN_fluxB + DeltaL * FHN_fluxL
     +                 - kDH * Hpool(site)

c calculate MF(i)

      do 11 i = 1,5

          if((100.*NFconc(2) - 1.16 * 100.*NFconc(1)) . le. 0.44) then
              MF(i) = 0.1
          else
              if((100.*NFconc(2) - 1.16*100.*NFconc(1)) . le. 1.50) then
                  MF(i) = 0.5
              else
                  MF(i) = 1.0
              endif
          endif
```

```
11    continue

          if((100.*NFconc(2) - 1.16 * 100.*NFconc(1)) . le. 0.44) then
              MF(11) = 0.1
          else
              if((100.*NFconc(2) - 1.16*100.*NFconc(1)) . le. 1.50) then
                  MF(11) = 0.5
              else
                  MF(11) = 1.0
              endif
          endif




c MF_flux is transferred from F to H, MF_min is released

      MF_flux = 0.
      MF_min = 0.

      do 20 i = 1,5


          d_LNpool(i) = litter_N(i) + litter_N(i+5)
     1                 - (ML*kDL(i) + kTL(i)) * LNpool(i,site)


          d_FNpool(i) = (kTL(i)) * LNpool(i,site)
     1                  - MF(i) * kDF(i) * FNpool(i,site)
     1                  -(kTF(i) + kTF2(i)) * FNpool(i,site)

          MF_flux = MF_flux + (1-MF(i)) * kDF(i) * FNpool(i,site)
          MF_min = MF_min + MF(i) * kDF(i) * FNpool(i,site)

          Lflux(i,site) = Lflux(i,site) + ML*kDL(i) * LNpool(i,site)
          Fflux(i,site) = Fflux(i,site) + MF(i)*kDF(i) * FNpool(i,site)



20    continue



          d_LNpool(11) = fn(11) * litter(11)
     1                 - (ML*kDL(11) + kTL(11)) * LNpool(11,site)


          d_FNpool(11) = (kTL(11)) * LNpool(11,site)
     1                  - MF(11) * kDF(11) * FNpool(11,site)
     1                  -(kTF(11) + kTF2(11)) * FNpool(11,site)
```

```
          MF_flux = MF_flux + (1-MF(11)) * kDF(11) * FNpool(11,site)
          MF_min = MF_min + MF(11) * kDF(11) * FNpool(11,site)



        if(HNpool(site) .gt. 0.) then
          H_CN = Hpool(site) / HNPool(site) / 2.
       else
          H_CN = 100.
       endif

       if(H_CN .gt. 8.) then
          MH = 0.8
       else
          MH = 1.
       endif


          d_HumusNPool =
      +                   - kDH * MH * HNpool(site)
      1                   + gamma * (FHN_fluxB + FHN_fluxL)



C  Calculate mineralised carbon

       H_miner = kDH * Hpool(site) + FH_fluxB + FH_fluxL
      1                  - DeltaB * FHN_fluxB - DeltaL * FHN_fluxL
      1                  - DeltaB * kM * Navail(site)

       do 15 i = 1,5
          H_miner = H_miner + kDL(i) * Lpool(i,site)
      +              + kDF(i) * Fpool(i,site)
15     continue

          H_miner = H_miner + kDL(11) * Lpool(11,site)
      +              + kDF(11) * Fpool(11,site)


       H_C = C_C * H_miner
       C_sum(site) = C_sum(site) + H_C

c calculate N release

       N_release =    kDH * MH * HNpool(site)
      1              + (1. - gamma) *(FHN_fluxB + FHN_fluxL)
      1                 + DeposN
c     1                 - kM * Navail(site) + DeposN
```

```
      Hflux(site) = Hflux(site) + kDH* MH * HNpool(site)


      do 16 i = 1,5
          N_release = N_release + ML * kDL(i) * LNpool(i,site)
     1                + MF(i) * kDF(i) * FNpool(i,site)


16    continue

          N_release = N_release + ML * kDL(11) * LNpool(11,site)
     1                + MF(11) * kDF(11) * FNpool(11,site)


       d_Navail = N_release  - LeachN
c     1             - N_coeff_TR * Ndemand
c     1             - N_coeff_GV1 * Gr_GVegN1 - N_coeff_GV2 *  Gr_GVegN2


c Update pools - use simple Euler

      do 17 i = 1, 5
c daily version -> step equals one!
          Lpool(i,site) = Lpool(i,site) + d_Lpool(i) * step
          Fpool(i,site) = Fpool(i,site) + d_Fpool(i) * step
          LNpool(i,site) = LNpool(i,site) + d_LNpool(i) * step
          FNpool(i,site) = FNpool(i,site) + d_FNpool(i) * step
17    continue

          Lpool(11,site) = Lpool(11,site) + d_Lpool(11) * step
          Fpool(11,site) = Fpool(11,site) + d_Fpool(11) * step
          LNpool(11,site) = LNpool(11,site) + d_LNpool(11) * step
          FNpool(11,site) = FNpool(11,site) + d_FNpool(11) * step


      HPool(site) = HPool(site) + d_HumusPool * step
       HNPool(site) = HNPool(site) + d_HumusNPool * step
       Navail(site) = Navail(site) + d_Navail * step

      N_sum(site) = N_sum(site) + d_Navail * step

c      Navail(site) = max(Navail(site), 0.)




      return
      end subroutine ROMUL
```

```
c       ------------------------------------------------------------------------
c       The functions for rate of decomposition modifiers depending on temperature
c       of the corresponding cohort. These new functions are from Komarov et al. 2008.
c
c       Tapio Linkosalo October 2008
c       ------------------------------------------------------------------------
        real function f_1(T)

                real T

                if (T .LE. -5.0 .OR. T .GT. 60.0) then
                        f_1 = 0
                endif
                if (T .GT. -5.0 .AND. T .LE. 1.0) then
                        f_1 = 0.1595 + 0.0319 * T
                endif
                if (T . GT. 1.0 .AND. T .LE. 35.0) then
                        f_1 = 0.1754 * exp(0.0871 * T)
                endif
                if (T . GT. 35.0 .AND. T .LE. 60.0) then
                        f_1 = 8.791 - 0.1465 * T
                endif

                return
        end
c       ------------------------------------------------------------------------
        real function f_2(T)

                real T

                if (T .LE. -5.0 .OR. T .GT. 60.0) then
                        f_2 = 0
                endif
                if (T .GT. -5.0 .AND. T .LE. 1.0) then
                        f_2 = 0.1595 + 0.0319 * T
                endif
                if (T . GT. 1.0 .AND. T .LE. 35.0) then
                        f_2 = 0.1754 * exp(0.0871 * T)
                endif
                if (T . GT. 35.0 .AND. T .LE. 60.0) then
                        f_2 = 3.690 - 0.0615 * T
                endif

                return
        end
c       ------------------------------------------------------------------------
        real function f_3(T)
```

```
              real T

              if (T .LE. -3.0) then
                     f_3 = 0
              endif
              if (T .GT. -3.0 .AND. T .LE. 7.0) then
                     f_3 = 1.3
              endif
              if (T .GT. 7.0 .AND. T .LE. 60.0) then
                     f_3 = 1.472 - T * 0.0245
              endif
              if (T. GT. 60.0) then
                     f_3 = 0
              endif

              return
       end
c      ------------------------------------------------------------------------
       real function f_4(T)

              real T

              if (T .LE. -5.0) then
                     f_4 = 0
              endif
              if (T .GT. -5.0 .AND. T .LE. 1.0) then
                     f_4 = 0.1595 + 0.0319 * T
              endif
              if (T .GT. 1.0 .AND. T .LE. 20.0) then
                     f_4 = 0.1754 * exp(0.0871 * T)
              endif
              if (T .GT. 20.0 .AND. T .LE. 40.0) then
                     f_4 = 1
              endif
              if (T .GT. 40.0 .AND. T .LE. 80.0) then
                     f_4 = 2.0 - 0.025 * T
              endif
              if (T. GT. 80.0) then
                     f_4 = 0
              endif

              return
       end
c      ------------------------------------------------------------------------
       real function f_5(T)

              real T

              if (T .LE. -5.0) then
```

```
                f_5 = 0
            endif
            if (T .GT. -5.0 .AND. T .LE. 1.0) then
                f_5 = 0.078 + 0.0156 * T
            endif
            if (T .GT. 1.0 .AND. T .LE. 13.0) then
                f_5 = 0.0675 * exp(0.2088 * T)
            endif
            if (T .GT. 13.0 .AND. T .LE. 25.0) then
                f_5 = 1
            endif
            if (T .GT. 25.0 .AND. T .LE. 50.0) then
                f_5 = 2.0 - 0.04 * T
            endif
            if (T. GT. 50.0) then
                f_5 = 0
            endif

            return
        end
c       ------------------------------------------------------------------------------
        real function f_6(T)

            real T

            if (T .LE. -5.0) then
                f_6 = 0
            endif
            if (T .GT. -5.0 .AND. T .LE. 1.0) then
                f_6 = 0.1595 + 0.0319 * T
            endif
            if (T .GT. 1.0 .AND. T .LE. 27.5) then
                f_6 = 0.1754 * exp(0.0871 * T)
            endif
            if (T .GT. 27.5 .AND. T .LE. 35.0) then
                f_6 = 1.95
            endif
            if (T .GT. 35.0 .AND. T .LE. 60.0) then
                f_6 = 4.68 - 0.078 * T
            endif
            if (T. GT. 60.0) then
                f_6 = 0
            endif

            return
        end
c       ------------------------------------------------------------------------------
c
c     Decomposition rate function for volumetric soil water content (theta) (where
c     theta = 0 == wilting point and theta = 1 == saturation), based on paper
```

```
c     Linkosalo, Kolari & Pumpanen 2013.
c
c      ------------------------------------------------------------------------
      real function LKP_decomp(theta, porosity)

         real  theta, porosity, P1, P2

         P1 = 3.83 * theta ** 1.25
         P2 = 4.43 * (1-theta)**0.8854
         LKP_decomp = min(P1,P2,1.)

         return
         end


c      ------------------------------------------------------------------------
c      The following are the k coefficients for the decomposition rate, depending on
c      litter ash and N content. The two values are given as parameters (absolute
c      values g/g, NOT percentage as in Romul equations!) so that the same functions
c      can be used whether the parameter values are for a specific cohort or litter
c      in general. These are the "new" functions as in Komarov et al. 2008.
c
c      TL October 2008.
c      ------------------------------------------------------------------------
       double precision function k_1L_rma(ash, N)

              double precision ash, N

              k_1L_rma = 0.0005 + 0.54 * N

              return
       end
c      ------------------------------------------------------------------------
       double precision function k_1S_rma(ash, N)

              double precision ash, N

              k_1S_rma = 0.0136 + 0.06 * ash

              return
       end
c      ------------------------------------------------------------------------
       double precision function k_2L_rma(ash, N)

              double precision ash, N

              k_2L_rma = 0.00060

              return
       end
```

```
c       ------------------------------------------------------------------------
        double precision function k_2S_rma(ash, N)

                double precision ash, N

                k_2S_rma = 0.00126

                return
        end
c       ------------------------------------------------------------------------
        double precision function k_3L_rma(ash, N)

                double precision ash, N


                k_3L_rma = 0.0089 + 0.78 * N

                return
        end
c       ------------------------------------------------------------------------
        double precision function k_3S_rma(ash, N)

                double precision ash, N


                if (ash .LT. 0.18) then
               k_3s_rma = 0.0394 - 0.21 * ash
                else
               k_3S_rma = 0.0394 - 0.21 * 0.18
                endif

                return
        end
c       ------------------------------------------------------------------------
        double precision function k_4_rma(ash, N)

                double precision ash, N

                if (N .LE. 0.02) then
                       k_4_rma = 0.05 * N
                else
                       k_4_rma = 0.001
                endif

        return
        end
c       ------------------------------------------------------------------------
        double precision function k_5_rma(ash, N)

                double precision ash, N
```

```
              if (N .LE. 0.005) then
                     k_5_rma = 0
              else
                     if (N .GE. 0.02) then
                            k_5_rma = 0.007
                     else
                            k_5_rma = 0.007 * (100*(2*N - 0.01)/3)
                     endif
              endif
       return
       end
c      ------------------------------------------------------------------------------
       double precision function k_6_rma()

              k_6_rma = 0.00006

       return
       end
c      ------------------------------------------------------------------------------


c ***************************************************************
c
c Subroutine simulates the soil water content in two layers,
c organic layer on top and mineral soil layer in bottom.
c
c Model presented in Linkosalo, Kolari and Pumpanen 2013.
c
c Input/output parameters: SW (SoilWater, absolute, in mm)
c                        theta (output for calc, 0 = WP and 1 = sat)
c                        prec (precipitation, in mm)
c                        ET_tot (total evapotranspiration in mm)
c
c Local parameters per layer: saturation (mm), FC (mm), WP (mm), tau (days)
c                           ET_ratio (split of ET between layers)
c
c ***************************************************************

      Subroutine two_layer_soil_water(SoilW, SWmax, theta, prec, ET_tot)

c local variables
       integer i
       real theta(2), SoilW(2), ET_tot, prec
         real FC(2), WP(2), saturation(2), ET(2)
       real tau_soil(2), ET_ratio, overflow
       real P1, P2, P_H, P_M
         real SWmax(2)

c     Soil water submodel parameters
```

```
        ET_ratio = 0.256107371
        tau_soil(1) = 0.894587365
        tau_soil(2) = 9.418136372
        saturation(1) = SWmax(1)/0.65
        saturation(2) = SWmax(2)/0.65
       wp(1) = 0
       wp(2) = 0
       FC(1) = WP(1) + SWmax(1)
       FC(2) = WP(2) + SWmax(2)

c Split evapotranspiration for the two layers
        ET(1) = ET_ratio * ET_tot
        if(ET(1) .gt. SoilW(1)) then
               ET(1) = SoilW(1)
        endif
        ET(2) = ET_tot - ET(1)


c Soilwater of organic layer from previous day over field capacity? -> overflow
        if (SoilW(1) .gt. FC(1)) then
               overflow = (SoilW(1) - FC(1)) / tau_soil(1)
               SoilW(1) = FC(1)
        else
               overflow = 0
        endif

        SoilW(1) = SoilW(1) + prec - ET(1)

c New soil water of organic layer over saturation -> immediately drainage
        if (SoilW(1) .gt. saturation(1)) then
               overflow = overflow + (SoilW(1) - saturation(1))
               SoilW(1) = saturation(1)
        endif

c           SoilW(2) = max(SoilW(2), SoilWP(2))

c Mineral soil over FC -> overflow
        if (soilW(2) .gt. FC(2)) then
               soilW(2) = soilW(2) - (soilW(2) - FC(2)) / tau_soil(2)
        endif

c Now add new water and subtract ET
        SoilW(2) = SoilW(2) + overflow - ET(2)

        if (soilW(2) .gt. saturation(2)) then
               soilW(2) = saturation(2)
c              overflow = saturation(2) - SoilW(2)
        endif

        if (SoilW(2) .lt. WP(2)) then
```

```
               ET(2) = ET(2) - (WP(2) - SoilW(2))
               soilW(2) = WP(2)
        endif

        ET_tot = ET(1) + ET(2)

        theta(1) = (soilW(1) - WP(1)) / (Saturation(1) - WP(1))
        theta(2) = (soilW(2) - WP(2)) / (Saturation(2) - WP(2))


        return
       end


c-------------------------------------------------------------------------
c
c       Evapotranspiration function based loosely on paper
c       Duursma et al. Tree Physiology 2008, but the dependency of ET on irradiation
c       modified by T. Linkosalo and fitted to Hyytiälä data in spring 2009
c
c-------------------------------------------------------------------------
        real function EvapoTranspiration (Temp, PAR, VPD, x, CO2effect,
     +          CO2ppm, REW, fDET)

c
        real ET
        real Temp, PAR, VPD, x, REW, fDET
        real beta, tau, x0, kappa, a_1, a_2, CO2ppm
        integer CO2effect

c       parameters hard-coded...

        beta = 0.016752
        tau = 14.39305
        x0 = -6.94684
        kappa = -0.000263
        a_1 = 0.0007
        a_2 = 0.0837


c       calculate S and D functions
               x = x + (Temp - x)/tau
               fS = max(0.0, x - x0)
               fD = exp(kappa * VPD)

c       calculate ET
               ET = beta * PAR * fS * fD + a_1*PAR + a_2

c Convert ET from mol/m2/d to g/m2/d
               ET = ET * 18
```

```
c Convert ET from g/m2/d to mm/d  (ET/rho and m -> mm)
               ET = ET / 1000

               fDET = 1

               if (REW .LT. 0.4) then
                      fDET = REW/0.4
                      ET = ET * fDET
               endif

               EvapoTranspiration = ET
               return
      end


c-------------------------------------------------------------------------
      subroutine SoilTemperature(ST, T)

      implicit none

      real ST(2), T, adj_T(2)

c parameters for soil temperature model
      real minimum_temp(2), tau(2)
      minimum_temp(1) = -0.13
      minimum_temp(2) = 0.24
       tau(1) = 14.9
       tau(2) = 10.5

      adj_T(1) = max(T, minimum_temp(1))
      adj_T(2) = max(T, minimum_temp(2))

      ST(1) = ST(1) + (adj_T(1) - ST(1)) / tau(1)
      ST(2) = ST(2) + (adj_T(2) - ST(2)) / tau(2)

      return
      end
```