Boreal Ecosystem Productivity Simulator Hourly v4.10

Generated by Doxygen 1.9.2

Chapter 1

BEPS hourly version (v4.10)

The user guide for BEPS hourly version for site (v4.10)

This model was initially developed for boreal ecosystems and has been adapted for all ecosystems over the globe.
BEPS mechanistically includes the impacts of various drivers on gross primary productivity (GPP) (climate, CO2 concentration, and nitrogen deposition) and assimilates vegetation structure (LAI) data.
BEPS also simulates the dynamics of carbon pools beyond GPP and uses a spin-up procedure to prescribe soil carbon pools for estimating autotrophic respiration (AR) and heterotrophic respiration (HR).
The BEPS hourly version for site (v4.10) can be used in two ways:
1) Dependency import
Please copy the header file and source file into traditional IDEs (i.e. Code::block, https://www.codeblocks.org/) and directly build and run the model.
2CMake
Please find the "CMakeLists.txt" file. The BEPS v4.10 model requires minimum 3.17 CMake version and is based on C99 standard.
It is recommended to use CLion (https://www.jetbrains.com/clion/) and MingW (https://www.mingw-w64.org/) to compile and run the model.
Make sure the "input" and "output" folders have been created in the current folder of the source codes.
According to users' research interests, the parameters and code structure can be edited. Please remember to make readable comment and git version control after each edition.
Please cite [ARTICLES] for using the BEPS model.
Please see "Modules_variables4BEPS.docx" for detailed parameter descriptions.

The BEPS model requires four input files: 1) Basic information; 2) Carbon pool data; 3) Leaf area index; 4) Meteorological data.
Users can find input data examples in the 'input' folder.
1) Basic information (data1 in the input data example)
long, lat, LC, CI, soiltxt, soiltemp, soilwater, snowdp [WITH TAB SPACE]
long – the longitude of site
lat – the latitude of site
LC – land cover type of site
1-ENF 2-DNF 6-DBF 9-EBF 13-shrub 40-C4 plants default-others
CI – clumping index
soiltxt – soil texture
1-land 2-loamy sand 3-sandy loam 4-loam 5-silty loam 6-sandy clay loam 7-clay loam 8-silty clay loam 9-sandy clay 10-silty clay 11-clay default-Others
soiltemp – soil temperature
soilwater – soil water content
snowdp – snow depth
2) Carbon pool data
LAI_yr, ann_NPP, ccd, cssd, csmd, cfsd, cfmd, csm, cm, cs, cp [WITH TAB SPACE]
3) Leaf area index
Daily float number LAI [WITH TAB SPACE]
4) Meteorological data

DOY, H, SW, TA, VPD/RH, P, WS [WITH TAB SPACE] [LINEBREAK EACH HOUR]
DOY – day of year (1-365)
H – hour of day (1-24)
SW – shortwave radiation
TA – air temperature
VPD/RH – vapor pressure deficit OR humidity
P – precipitation
WS – wind speed

References for algorithms in this model

He, L.; Wang, R.; Mostovoy, G.; Liu, J.; Chen, J.M.; Shang, J.; Liu, J.; McNairn, H.; Powers, J. Crop Biomass Mapping Based on Ecosystem Modeling at Regional Scale Using High Resolution Sentinel-2 Data. Remote Sens. 2021, 13, 806. https://doi.org/10.3390/rs13040806

Luo, X., Keenan, T.F. Global evidence for the acclimation of ecosystem photosynthesis to light. Nat Ecol Evol (2020). https://doi.org/10.1038/s41559-020-1258-7

Chen, J.M., Ju, W., Ciais, P. et al. Vegetation structural change since 1981 significantly enhanced the terrestrial carbon sink. Nat Commun 10, 4259 (2019). https://doi.org/10.1038/s41467-019-12257-8 (for daily version)

He, L., et al. (2019). "Diverse photosynthetic capacity of global ecosystems mapped by satellite chlorophyll fluorescence measurements." Remote Sensing of Environment 232: 111344.

He, L.; Mostovoy, G. Cotton Yield Estimate Using Sentinel-2 Data and an Ecosystem Model over the Southern US. Remote Sens. 2019, 11, 2000.

Luo, X, Croft, H, Chen, JM, He, L, Keenan, TF. Improved estimates of global terrestrial photosynthesis using information on leaf chlorophyll content. Glob Change Biol. 2019; 25: 2499– 2514. https://doi.org/10.1111/gcb.14624

Luo, X. Z., Chen, J. M., Liu, J. E., Black, T. A., Croft, H., Staebler, R., . . . McCaughey, H. (2018). Comparison of Big-Leaf, Two-Big-Leaf, and Two-Leaf Upscaling Schemes for Evapotranspiration Estimation Using Coupled Carbon-Water Modeling. Journal of Geophysical Research-Biogeosciences, 123(1), 207-225.

He, L. M., Chen, J. M., Gonsamo, A., Luo, X. Z., Wang, R., Liu, Y., & Liu, R. G. (2018). Changes in the Shadow: The Shifting Role of Shaded Leaves in Global Carbon and Water Cycles Under Climate Change. Geophysical Research Letters, 45(10), 5052-5061.

He, L. M., Chen, J. M., Croft, H., Gonsamo, A., Luo, X. Z., Liu, J. N., . . . Liu, Y. (2017). Nitrogen Availability Dampens the Positive Impacts of CO2 Fertilization on Terrestrial Ecosystem Carbon and Water Cycles. Geophysical Research Letters, 44(22), 11590-11600. doi:10.1002/2017gl075981

He, L., Chen, J. M., Liu, J., Bélair, S., & Luo, X. (2017). Assessment of SMAP soil moisture for global simulation of gross primary production. Journal of Geophysical Research: Biogeosciences, 122, doi:10.1002/2016JG003603. doi:10.1002/2016JG003603

Chen, B., Liu, J., Chen, J. M., Croft, H., Gonsamo, A., He, L., & Luo, X. (2016). Assessment of foliage clumping effects on evapotranspiration estimates in forested ecosystems. Agricultural and Forest Meteorology, 216, 82-92. doi:http://dx.doi.org/10.1016/j.agrformet.2015.09.017

He, L., Chen, J. M., Liu, J., Mo, G., Bélair, S., Zheng, T., . . . Barr, A. G. (2014). Optimization of water uptake and photosynthetic parameters in an ecosystem model using tower flux data. Ecological Modelling, 294(0), 94-104. doi:http://dx.doi.org/10.1016/j.ecolmodel.2014.09.019

Chen, J. M., Mo, G., Pisek, J., Liu, J., Deng, F., Ishizawa, M., & Chan, D. (2012). Effects of foliage clumping on the estimation of global terrestrial gross primary productivity. Global Biogeochemical Cycles, 26. doi:Artn Gb1019 Doi 10.1029/2010gb003996

Chen, B. Z., Chen, J. M., & Ju, W. M. (2007). Remote sensing-based ecosystem-atmosphere simulation scheme (EASS) - Model formulation and test with multiple-year data. Ecological Modelling, 209(2-4), 277-300. doi:DOI 10.1016/j.ecolmodel.2007.06.032

Ju, W., Chen, J. M., Black, T. A., Barr, A. G., Liu, J., & Chen, B. (2006). Modelling multi-year coupled carbon and water fluxes in a boreal aspen forest. Agricultural and Forest Meteorology, 140(1-4), 136-151. doi:10.1016/j.agrformet.2006.08.008

Chen, J. M., Liu, J., Cihlar, J., & Goulden, M. L. (1999). Daily canopy photosynthesis model through temporal and spatial scaling for remote sensing applications. Ecological Modelling, 124(2-3), 99-119. doi:Doi 10.1016/S0304-3800(99)00156-8

Liu, J., Chen, J. M., Cihlar, J., & Park, W. M. (1997). A process-based boreal ecosystem productivity simulator using remote sensing inputs. Remote Sensing of Environment, 62(2), 158-175.

Compiled into doxygen format by Jiye Leng @UofT
Oct., 2021

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 boundary_layer_resistances Struct Reference

Public Attributes

- double **vapor**
- double **heat**
- double **co2**

The documentation for this struct was generated from the following file:

- DB.h

4.2 climatedata Struct Reference

Declare structures.

```
#include <beps.h>
```

Public Attributes

- double **Srad**
- double **LR**
- double **temp**
- double **rh**
- double **rain**
- double **wind**
- double **dr_o**
- double **df_o**
- double **dr_u**
- double **df_u**

4.2.1 Detailed Description

Declare structures.

The documentation for this struct was generated from the following file:

- beps.h

4.3 cpools Struct Reference

Public Attributes

- double **Ccd** [3]

- double **Cssd** [3]
- double **Csmd** [3]
- double **Cfsd** [3]
- double **Cfmd** [3]
- double **Csm** [3]
- double **Cm** [3]
- double **Cs** [3]
- double **Cp** [3]

The documentation for this struct was generated from the following file:

- beps.h

4.4 factors Struct Reference

Public Attributes

- double **latent**
- double **latent18**
- double **heatcoef**
- double **a_filt**
- double **b_filt**
- double **co2**

The documentation for this struct was generated from the following file:

- DB.h

4.5 meteorology Struct Reference

Public Attributes

- double **ustar**
- double **ustarnew**
- double **rhova_g**
- double **rhova_kg**
- double **sensible_heat_flux**
- double **H_old**
- double **air_density**
- double **T_Kelvin**
- double **press_kpa**
- double **press_bars**
- double **press_Pa**
- double **pstat273**
- double **air_density_mole**
- double **relative_humidity**
- double **vpd**
- double **ir_in**

The documentation for this struct was generated from the following file:

- DB.h

4.6 results Struct Reference

Public Attributes

- double **gpp_o_sunlit**
- double **gpp_u_sunlit**
- double **gpp_o_shaded**
- double **gpp_u_shaded**
- double **plant_resp**
- double **npp_o**
- double **npp_u**
- double **GPP**
- double **NPP**
- double **NEP**
- double **soil_resp**
- double **Net_Rad**
- double **SH**
- double **LH**
- double **Trans**
- double **Evap**

The documentation for this struct was generated from the following file:

- beps.h

4.7 Soil Struct Reference

Define soil struct.
`#include <soil.h>`

Public Attributes

- int **flag**
- int **n_layer**
- int **step_period**
- double **Zp**
- double **Zsp**
- double **r_rain_g**
- double **soil_r**
- double **r_drainage**
- double **r_root_decay**
- double **psi_min**
- double **alpha**
- double **f_soilwater**
- double **d_soil** [MAX_LAYERS]
- double **f_root** [MAX_LAYERS]
- double **dt** [MAX_LAYERS]
- double **thermal_cond** [MAX_LAYERS]
- double **theta_vfc** [MAX_LAYERS]
- double **theta_vwp** [MAX_LAYERS]
- double **fei** [MAX_LAYERS]
- double **Ksat** [MAX_LAYERS]
- double **psi_sat** [MAX_LAYERS]
- double **b** [MAX_LAYERS]
- double **density_soil** [MAX_LAYERS]
- double **f_org** [MAX_LAYERS]

- double **ice_ratio** [MAX_LAYERS]
- double **thetam** [MAX_LAYERS]
- double **thetam_prev** [MAX_LAYERS]
- double **temp_soil_p** [MAX_LAYERS]
- double **temp_soil_c** [MAX_LAYERS]
- double **f_ice** [MAX_LAYERS]
- double **psim** [MAX_LAYERS]
- double **thetab** [MAX_LAYERS]
- double **psib** [MAX_LAYERS]
- double **r_waterflow** [MAX_LAYERS]
- double **km** [MAX_LAYERS]
- double **Kb** [MAX_LAYERS]
- double **KK** [MAX_LAYERS]
- double **Cs** [MAX_LAYERS]
- double **lambda** [MAX_LAYERS]
- double **Ett** [MAX_LAYERS]
- double **G** [MAX_LAYERS]

4.7.1 Detailed Description

Define soil struct.
The documentation for this struct was generated from the following file:

- soil.h

Chapter 5

File Documentation

5.1 aerodynamic_conductance.c File Reference

Calculation of aerodynamic resistance/conductance.

```
#include "beps.h"
```

Functions

- void aerodynamic_conductance (double canopy_height_o, double canopy_height_u, double zz, double clumping, double temp_air, double wind_sp, double SH_o_p, double lai_o, double lai_u, double ∗rm, double ∗ra_u, double ∗ra_g, double ∗G_o_a, double ∗G_o_b, double ∗G_u_a, double ∗G_u_b)

  Function to calculate aerodynamic resistance and conductance.

5.1.1 Detailed Description

Calculation of aerodynamic resistance/conductance.

Authors

Written by: J. Liu and W. Ju

Modified by G. Mo

Date

Last update: May 2015

5.1.2 Function Documentation

5.1.2.1 aerodynamic_conductance()

```
void aerodynamic_conductance (
            double canopy_height_o,
            double canopy_height_u,
            double zz,
            double clumping,
            double temp_air,
            double wind_sp,
            double SH_o_p,
            double lai_o,
            double lai_u,
            double * rm,
            double * ra_u,
```

```
double * ra_g,
double * G_o_a,
double * G_o_b,
double * G_u_a,
double * G_u_b )
```

Function to calculate aerodynamic resistance and conductance.

Parameters

canopy_height←↩_o	canopy height, overstory
canopy_height←↩_u	height of understory
zz	the height to measure wind speed
clumping	clumping index
temp_air	air temperature
wind_sp	wind speed
SH_o_p	sensible heat flux from overstory
lai_o	leaf area index, overstory (lai_o+stem_o)
lai_u	leaf area index, understory (lai_u+stem_u)
rm	aerodynamic resistance, overstory, in s/m
ra_u	aerodynamic resistance, understory, in s/m
ra_g	aerodynamic resistance, ground, in s/m
G_o_a	aerodynamic conductance for leaves, overstory
G_o_b	boundary layer conductance for leaves, overstory
G_u_a	aerodynamic conductance for leaves, understory
G_u_b	boundary layer conductance for leaves, understory

Returns

void

5.2 beps.h File Reference

Header file for defining constants and global variables for BEPS program.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include "soil.h"
```

Classes

- struct climatedata

  Declare structures.
- struct results
- struct cpools

Macros

- #define **NOERROR** 0

  Define Constants.
- #define **ERROR** 1

- #define **PI** 3.1415926
- #define **zero** 0.0000000001
- #define **max**(a, b) ((a)>(b))?(a):(b)
- #define **min**(a, b) ((a)<(b))?(a):(b)
- #define **l_sta** 105
- #define **l_end** 105
- #define **p_sta** 101
- #define **p_end** 101
- #define **RTIMES** 24
- #define **step** 3600
- #define **kstep** 360
- #define **kloop** 10
- #define **layer** 5
- #define **depth_f** 6
- #define **CO2_air** 380
- #define **rho_a** 1.292

Functions

- void **readconf** ()

    Declare functions.
- void **mid_prg** ()
- void **readinput1** ()
- void **readlai_d** ()
- void **readlonlat** ()
- void inter_prg (int jday, int rstep, double lai, double clumping, double parameter[ ], struct climatedata ∗meteo, double CosZs, double var_o[ ], double var_n[ ], struct Soil ∗soilp, struct results ∗mid_res)

    the inter-module function between main program and modules
- void s_coszs (short jday, short j, float lat, float lon, double ∗CosZs)

    Function to calculate cosine solar zenith angle.
- void aerodynamic_conductance (double canopy_height_o, double canopy_height_u, double zz, double clumping, double temp_air, double wind_sp, double SH_o_p, double lai_o, double lai_u, double ∗rm, double ∗ra_u, double ∗ra_g, double ∗G_o_a, double ∗G_o_b, double ∗G_u_a, double ∗G_u_b)

    Function to calculate aerodynamic resistance and conductance.
- void plantresp (int LC, struct results ∗mid_res, double lai_yr, double lai, double temp_air, double temp_soil, double CosZs)

    Function to calculate plant respiration.
- void Vcmax_Jmax (double lai_o, double clumping, double Vcmax0, double slope_Vcmax_N, double leaf_N, double CosZs, double ∗Vcmax_sunlit, double ∗Vcmax_shaded, double ∗Jmax_sunlit, double ∗Jmax_shaded)

    Function to calculate the Vcmax and Jmax for sunlit and shaded leaf.
- void netRadiation (double shortRad_global, double CosZs, double temp_o, double temp_u, double temp_g, double lai_o, double lai_u, double lai_os, double lai_us, double lai_o_sunlit, double lai_o_shaded, double lai_u_sunlit, double lai_u_shaded, double clumping, double temp_air, double rh, double albedo_snow_↩ v, double albedo_snow_n, double percentArea_snow_o, double percentArea_snow_u, double percent_↩ snow_g, double albedo_v_o, double albedo_n_o, double albedo_v_u, double albedo_n_u, double albedo_↩ v_g, double albedo_n_g, double ∗netRad_o, double ∗netRad_u, double ∗netRad_g, double ∗netRadLeaf↩ _o_sunlit, double ∗netRadLeaf_o_shaded, double ∗netRadLeaf_u_sunlit, double ∗netRadLeaf_u_shaded, double ∗netShortRadLeaf_o_sunlit, double ∗netShortRadLeaf_o_shaded, double ∗netShortRadLeaf_u_↩ sunlit, double ∗netShortRadLeaf_u_shaded)

    Function to calculate net radiation at canopy level and leaf level.
- void soilresp (double ∗Ccd, double ∗Cssd, double ∗Csmd, double ∗Cfsd, double ∗Cfmd, double ∗Csm, double ∗Cm, double ∗Cs, double ∗Cp, float npp_yr, double ∗coef, int soiltype, struct Soil ∗soilp, struct results ∗mid↩ _res)

    Function to calculate soil respiration.

- void **readparam** (short lc, double parameter1[ ])
- void lai2 (double stem_o, double stem_u, int LC, double CosZs, double lai_o, double clumping, double lai_u, double ∗lai_o_sunlit, double ∗lai_o_shaded, double ∗lai_u_sunlit, double ∗lai_u_shaded, double ∗PAI_o_↩ sunlit, double ∗PAI_o_shaded, double ∗PAI_u_sunlit, double ∗PAI_u_shaded)

  Function to recalculate sunlit and shaded leaf area index.
- void **readcoef** (short lc, int stxt, double coef[ ])
- void **readhydr_param** ()
- void photosynthesis (double temp_leaf_p, double rad_leaf, double e_air, double g_lb_w, double vc_opt, double f_soilwater, double b_h2o, double m_h2o, double cii, double temp_leaf_c, double LH_leaf, double ∗Gs_w, double ∗aphoto, double ∗ci)

  Function to calculate leaf photosynthesis by solving a cubic equation.
- void **soil_water_factor** ()
- void Leaf_Temperatures (double Tair, double slope, double psychrometer, double VPD_air, double Cp_ca, double Gw_o_sunlit, double Gw_o_shaded, double Gw_u_sunlit, double Gw_u_shaded, double Gww_o_↩ sunlit, double Gww_o_shaded, double Gww_u_sunlit, double Gww_u_shaded, double Gh_o_sunlit, double Gh_o_shaded, double Gh_u_sunlit, double Gh_u_shaded, double Xcs_o, double Xcl_o, double Xcs_u, double Xcl_u, double radiation_o_sun, double radiation_o_shaded, double radiation_u_sun, double radiation_↩ u_shaded, double ∗Tc_o_sunlit, double ∗Tc_o_shaded, double ∗Tc_u_sunlit, double ∗Tc_u_shaded)

  Function to calculate leaf temperature four components (sunlit and shaded leaves, overstory and understory)
- double Leaf_Temperature (double Tair, double slope, double psychrometer, double VPD_air, double Cp_ca, double Gw, double Gww, double Gh, double Xcs, double Xcl, double radiation)

  Subroutine to calculate leaf temperature.
- void sensible_heat (double tempL_o_sunlit, double tempL_o_shaded, double tempL_u_sunlit, double temp↩ L_u_shaded, double temp_g, double temp_air, double rh_air, double Gheat_o_sunlit, double Gheat_o_↩ shaded, double Gheat_u_sunlit, double Gheat_u_shaded, double Gheat_g, double lai_o_sunlit, double lai↩ _o_shaded, double lai_u_sunlit, double lai_u_shaded, double ∗SH_o, double ∗SH_u, double ∗SH_g)

  Function to calculate sensible heat.
- void transpiration (double tempL_o_sunlit, double tempL_o_shaded, double tempL_u_sunlit, double temp↩ L_u_shaded, double temp_air, double rh_air, double Gtrans_o_sunlit, double Gtrans_o_shaded, double Gtrans_u_sunlit, double Gtrans_u_shaded, double lai_o_sunlit, double lai_o_shaded, double lai_u_sunlit, double lai_u_shaded, double ∗trans_o, double ∗trans_u)

  Function to calculate transpiration.
- void evaporation_canopy (double tempL_o_sunlit, double tempL_o_shaded, double tempL_u_sunlit, double tempL_u_shaded, double temp_air, double rh_air, double Gwater_o_sunlit, double Gwater_o_shaded, double Gwater_u_sunlit, double Gwater_u_shaded, double lai_o_sunlit, double lai_o_shaded, double lai_u↩ _sunlit, double lai_u_shaded, double percent_water_o, double percent_water_u, double percent_snow_o, double percent_snow_u, double ∗evapo_water_o, double ∗evapo_water_u, double ∗evapo_snow_o, double ∗evapo_snow_u)

  Function to calculate evaporation and sublimation from canopy.
- void evaporation_soil (double temp_air, double temp_g, double rh_air, double netRad_g, double Gheat↩ _g, double ∗percent_snow_g, double ∗depth_water, double ∗depth_snow, double ∗mass_water_g, double ∗mass_snow_g, double density_snow, double swc_g, double porosity_g, double ∗evapo_soil, double ∗evapo_water_g, double ∗evapo_snow_g)

  Function to calculate evaporation from ground surface/top soil, and the evaporation of snow and pond water on surface.
- void rainfall_stage1 (double temp_air, double precipitation, double mass_water_o_last, double mass_water↩ _u_last, double lai_o, double lai_u, double clumping, double ∗mass_water_o, double ∗mass_water_u, double ∗percent_water_o, double ∗percent_water_u, double ∗precipitation_g)

  Function of rainfall stage1.
- void rainfall_stage2 (double evapo_water_o, double evapo_water_u, double ∗mass_water_o, double ∗mass↩ _water_u)

  Function of rainfall stage2.
- void **rainfall_stage3** ()
- void meteo_pack (double temp, double rh, double ∗meteo_pack_output)

  Function to calculate meteorological variables based on input.

- void surface_temperature (double temp_air, double rh_air, double depth_snow, double depth_water, double capacity_heat_soil1, double capacity_heat_soil0, double Gheat_g, double depth_soil1, double density_↩ snow, double tempL_u, double netRad_g, double evapo_soil, double evapo_water_g, double evapo_snow↩ _g, double lambda_soil1, double percent_snow_g, double heat_flux_soil1, double temp_ground_last, double temp_soil1_last, double temp_any0_last, double temp_snow_last, double temp_soil0_last, double temp↩ _snow1_last, double temp_snow2_last, double ∗temp_ground, double ∗temp_any0, double ∗temp_snow, double ∗temp_soil0, double ∗temp_snow1, double ∗temp_snow2, double ∗heat_flux)

    Function to simulate surface temperature, and heat flux from surface to soil layers.
- void snowpack_stage1 (double temp_air, double precipitation, double mass_snow_o_last, double mass_↩ snow_u_last, double mass_snow_g_last, double ∗mass_snow_o, double ∗mass_snow_u, double ∗mass↩ _snow_g, double lai_o, double lai_u, double clumping, double ∗area_snow_o, double ∗area_snow_u, double ∗percent_snow_o, double ∗percent_snow_u, double ∗percent_snow_g, double ∗density_snow, double ∗depth_snow, double ∗albedo_v_snow, double ∗albedo_n_snow)

    Function of snowpack stage1.
- void snowpack_stage2 (double evapo_snow_o, double evapo_snow_u, double ∗mass_snow_o, double ∗mass_snow_u)

    Function of snowpack stage2. This module will calculate the snow remained on canopy surface after evaporation in this step.
- void snowpack_stage3 (double temp_air, double temp_snow, double temp_snow_last, double density_snow, double ∗depth_snow, double ∗depth_water, double ∗mass_snow_g)

    Function of snowpack stage3. This module simulates the process of snow melting and water frozen in this step.

Variables

- short **lc_no**

    Declare global variables.
- int **yr**
- int **bgn_day**
- int **end_day**
- int **npixels**
- int **nlines**
- char **lc_fn** [255]
- char **lai_fn** [255]
- char **lai_fp** [255]
- char **stxt_fn** [255]
- char **ci_fn** [255]
- char **st_fn** [255]
- char **sw_fn** [255]
- char **sdp_fn** [255]
- char **r_fn** [255]
- char **t_fn** [255]
- char **h_fn** [255]
- char **p_fn** [255]
- char **wd_fn** [255]
- char **lon_fn** [255]
- char **lat_fn** [255]
- char **fp4outp1** [255]
- char **fp4outp2** [255]
- char **fp4outp3** [255]

5.2.1 Detailed Description

Header file for defining constants and global variables for BEPS program.
CCRS (EMS/Applications Division)

Author

Written by: J. Liu, Modified by: G. Mo

Date

June 2015

5.2.2 Function Documentation

5.2.2.1 aerodynamic_conductance()

```
void aerodynamic_conductance (
            double canopy_height_o,
            double canopy_height_u,
            double zz,
            double clumping,
            double temp_air,
            double wind_sp,
            double SH_o_p,
            double lai_o,
            double lai_u,
            double * rm,
            double * ra_u,
            double * ra_g,
            double * G_o_a,
            double * G_o_b,
            double * G_u_a,
            double * G_u_b )
```

Function to calculate aerodynamic resistance and conductance.

Parameters

canopy_height_o	canopy height, overstory
canopy_height_u	height of understory
zz	the height to measure wind speed
clumping	clumping index
temp_air	air temperature
wind_sp	wind speed
SH_o_p	sensible heat flux from overstory
lai_o	leaf area index, overstory (lai_o+stem_o)
lai_u	leaf area index, understory (lai_u+stem_u)
rm	aerodynamic resistance, overstory, in s/m
ra_u	aerodynamic resistance, understory, in s/m
ra_g	aerodynamic resistance, ground, in s/m
G_o_a	aerodynamic conductance for leaves, overstory
G_o_b	boundary layer conductance for leaves, overstory
G_u_a	aerodynamic conductance for leaves, understory
G_u_b	boundary layer conductance for leaves, understory

Returns

void

5.2.2.2 evaporation_canopy()

```
void evaporation_canopy (
            double tempL_o_sunlit,
            double tempL_o_shaded,
            double tempL_u_sunlit,
            double tempL_u_shaded,
            double temp_air,
            double rh_air,
            double Gwater_o_sunlit,
            double Gwater_o_shaded,
            double Gwater_u_sunlit,
            double Gwater_u_shaded,
            double lai_o_sunlit,
            double lai_o_shaded,
            double lai_u_sunlit,
            double lai_u_shaded,
            double percent_water_o,
            double percent_water_u,
            double percent_snow_o,
            double percent_snow_u,
            double * evapo_water_o,
            double * evapo_water_u,
            double * evapo_snow_o,
            double * evapo_snow_u )
```

Function to calculate evaporation and sublimation from canopy.
[input] temperature of sunlit and shaded leaves from other storey (leaf temperature module); temperature of air; relative humidity; aerodynamic conductance of water (snow) for sunlit shaded leaves from overstorey and understorey; percentage of overstorey or understorey covered by water or snow; leaf area index, sunlit and shaded, overstorey and understorey (from leaf area index module);
[output] evaporation of water and snow from overstorey and understorey

Parameters

tempL_o_sunlit	temperature of leaves, overstory, sunlit (leaf temperature module)
tempL_o_shaded	temperature of leaves, overstory, shaded
tempL_u_sunlit	temperature of leaves, understory, sunlit
tempL_u_shaded	temperature of leaves, understory, shaded
temp_air	air temperature
rh_air	relative humidity
Gwater_o_sunlit	aerodynamic conductance of water (snow) for overstory, sunlit leaves
Gwater_o_shaded	aerodynamic conductance of water (snow) for overstory, shaded leaves
Gwater_u_sunlit	aerodynamic conductance of water (snow) for understory, sunlit leaves
Gwater_u_shaded	aerodynamic conductance of water (snow) for understory, shaded leaves
lai_o_sunlit	leaf area index, overstory, sunlit (from leaf area index module)
lai_o_shaded	leaf area index, overstory, shaded
lai_u_sunlit	leaf area index, understory, sunlit
lai_u_shaded	leaf area index, understory, shaded
percent_water_o	percentage of overstorey covered by water
percent_water_u	percentage of understorey covered by water
percent_snow_o	percentage of overstorey covered by snow

Parameters

percent_snow_u	percentage of understorey covered by snow
evapo_water_o	evaporation of water from overstorey
evapo_water_u	evaporation of water from understorey
evapo_snow_o	evaporation of snow from overstorey
evapo_snow_u	evaporation of snow from understorey

Returns

void

5.2.2.3 evaporation_soil()

```
void evaporation_soil (
            double temp_air,
            double temp_g,
            double rh_air,
            double netRad_g,
            double Gheat_g,
            double * percent_snow_g,
            double * depth_water,
            double * depth_snow,
            double * mass_water_g,
            double * mass_snow_g,
            double density_snow,
            double swc_g,
            double porosity_g,
            double * evapo_soil,
            double * evapo_water_g,
            double * evapo_snow_g )
```

Function to calculate evaporation from ground surface/top soil, and the evaporation of snow and pond water on surface.
[input] air temperature; ground surface temperature; relative humidity of ground (BEPS takes it as the air RH); percentage of snow cover on ground; depth of water; depth of snow soil water content on first soil layer; porosity of first soil layer
[output] evaporation from soil surface; depth of water and snow on ground after evaporation and sublimation

Parameters

temp_air	air temperature
temp_g	ground temperature
rh_air	relative humidity of air
netRad_g	net radiation on ground
Gheat_g	aerodynamic conductance of heat on ground surface
percent_snow←↩ _g	percentage of snow on ground
depth_water	depth of water on ground, after rainfall and snowfall stage 1, before evaporation. output after subtracting evaporation
depth_snow	depth of snow on ground, ...
mass_water_g	mass of water on ground, output after subtracting evaporation
mass_snow_g	mass of snow on ground, ...
density_snow	density of snow, from snowpack stage1
swc_g	soil water content (from last step)

Parameters

porosity_g	porosity on ground
evapo_soil	evaporation from soil
evapo_water_g	evaporation from pond water
evapo_snow_g	evaporation from snow on surface

Returns

void

5.2.2.4 inter_prg()

```
void inter_prg (
            int jday,
            int rstep,
            double lai,
            double clumping,
            double parameter[ ],
            struct climatedata * meteo,
            double CosZs,
            double var_o[ ],
            double var_n[ ],
            struct Soil * soilp,
            struct results * mid_res )
```

the inter-module function between main program and modules

Parameters

jday	day of year
rstep	hour of day
lai	leaf area index
clumping	clumping index
parameter	parameter array according to land cover types
meteo	meteorological data
CosZs	cosine of solar zenith angle
var_o	temporary variables array of last time step
var_n	temporary variables array of this time step
soilp	soil coefficients according to land cover types and soil textures
mid_res	results struct

Returns

void

5.2.2.5 lai2()

```
void lai2 (
            double stem_o,
            double stem_u,
            int LC,
            double CosZs,
```

```
double lai_o,
double clumping,
double lai_u,
double * lai_o_sunlit,
double * lai_o_shaded,
double * lai_u_sunlit,
double * lai_u_shaded,
double * PAI_o_sunlit,
double * PAI_o_shaded,
double * PAI_u_sunlit,
double * PAI_u_shaded )
```

Function to recalculate sunlit and shaded leaf area index.

Parameters

stem_o	overstory woody area
stem_u	understory woody area
LC	land cover type
CosZs	cosine solar zenith angle
lai_o	overstory lai
clumping	clumping index
lai_u	understory lai
lai_o_sunlit	overstory sunlit lai
lai_o_shaded	overstory shaded lai
lai_u_sunlit	understory sunlit lai
lai_u_shaded	understory shaded lai
PAI_o_sunlit	overstory sunlit lai
PAI_o_shaded	overstory shaded lai
PAI_u_sunlit	understory sunlit lai
PAI_u_shaded	understory shaded lai

Returns

void

5.2.2.6 Leaf_Temperature()

```
double Leaf_Temperature (
        double Tair,
        double slope,
        double psychrometer,
        double VPD_air,
        double Cp_ca,
        double Gw,
        double Gww,
        double Gh,
        double Xcs,
        double Xcl,
        double radiation )
```

Subroutine to calculate leaf temperature.

Parameters

Tair	air temperature

Parameters

slope	the slope of saturation vapor pressure-temperature curve
psychrometer	psychrometer constant, 0.066 kPa K
VPD_air	vapor pressure deficit
Cp_ca	specific heat of moist air in kJ/kg/K
Gw	total conductance for water from the intercellular space of the leaves to the reference height above the canopy
Gww	total conductance for water from the surface of the leaves to the reference height above the canopy
Gh	total conductance for heat transfer from the leaf surface to the reference height above the canopy
Xcs	the fraction of canopy covered by snow
Xcl	the fraction of canopy covered by liquid water
radiation	net radiation on leaves

Returns

[double Tc] the effective canopy temperature in Kalvin

5.2.2.7 Leaf_Temperatures()

```
void Leaf_Temperatures (
            double Tair,
            double slope,
            double psychrometer,
            double VPD_air,
            double Cp_ca,
            double Gw_o_sunlit,
            double Gw_o_shaded,
            double Gw_u_sunlit,
            double Gw_u_shaded,
            double Gww_o_sunlit,
            double Gww_o_shaded,
            double Gww_u_sunlit,
            double Gww_u_shaded,
            double Gh_o_sunlit,
            double Gh_o_shaded,
            double Gh_u_sunlit,
            double Gh_u_shaded,
            double Xcs_o,
            double Xcl_o,
            double Xcs_u,
            double Xcl_u,
            double radiation_o_sun,
            double radiation_o_shaded,
            double radiation_u_sun,
            double radiation_u_shaded,
            double * Tc_o_sunlit,
            double * Tc_o_shaded,
            double * Tc_u_sunlit,
            double * Tc_u_shaded )
```

Function to calculate leaf temperature four components (sunlit and shaded leaves, overstory and understory) [output] Tc_o_sunlit,Tc_o_shaded,Tc_u_sunlit,Tc_u_shaded

Parameters

Tair	air temperature
slope	the slope of saturation vapor pressure-temperature curve
psychrometer	psychrometer constant, 0.066 kPa K
VPD_air	vapor pressure deficit
Cp_ca	specific heat of moist air in kJ/kg/K
Gw_o_sunlit	total conductance for water from the intercellular space of the leaves to the reference height above the canopy, overstory, sunlit
Gw_o_shaded	..., overstory, shaded
Gw_u_sunlit	..., understory, sunlit
Gw_u_shaded	..., understory, shaded
Gww_o_sunlit	total conductance for water from the surface of the leaves to the reference height above the canopy, overstory, sunlit
Gww_o_shaded	..., overstory, shaded
Gww_u_sunlit	..., understory, sunlit
Gww_u_shaded	..., understory, shaded
Gh_o_sunlit	total conductance for heat transfer from the leaf surface to the reference height above the canopy, overstory, sunlit
Gh_o_shaded	..., overstory, shaded
Gh_u_sunlit	..., understory, sunlit
Gh_u_shaded	..., understory, shaded
Xcs_o	the fraction of canopy covered by snow, overstory
Xcl_o	the fraction of canopy covered by liquid water, overstory
Xcs_u	the fraction of canopy covered by snow, understory
Xcl_u	the fraction of canopy covered by liquid water, understory
radiation_o_sun	net radiation on leaves, overstory, sunlit
radiation_o_shaded	net radiation on leaves, overstory, shaded
radiation_u_sun	net radiation on leaves, understory, sunlit
radiation_u_shaded	net radiation on leaves, understory, shaded
Tc_o_sunlit	the effective canopy temperature in Kalvin, overstory, sunlit
Tc_o_shaded	the effective canopy temperature in Kalvin, overstory, shaded
Tc_u_sunlit	the effective canopy temperature in Kalvin, understory, sunlit
Tc_u_shaded	the effective canopy temperature in Kalvin, understory, shaded

Returns

void

5.2.2.8 meteo_pack()

```
void meteo_pack (
            double temp,
            double rh,
            double * meteo_pack_output )
```

Function to calculate meteorological variables based on input.
default input is temperature (C) and relative humidity (0-100) output is an array, named as meteo_pack_output []
[input] meteo_pack_output [1]= air_density kg/m3
meteo_pack_output [2]= specific heat of air J/kg/C
meteo_pack_output [3]= VPD kPa
meteo_pack_output [4]= slope of vapor pressure to temperature kPa/C

meteo_pack_output [5]= psychrometer constant kPa/C
meteo_pack_output [6]= saturate water vapor potential kPa
meteo_pack_output [7]= actual water vapor potential kPa
meteo_pack_output [8]= specific humidity g/g

Parameters

temp	temperature
rh	relative humidity
meteo_pack_output	meteorological variables array

Returns

void

5.2.2.9 netRadiation()

```
void netRadiation (
            double shortRad_global,
            double CosZs,
            double temp_o,
            double temp_u,
            double temp_g,
            double lai_o,
            double lai_u,
            double lai_os,
            double lai_us,
            double lai_o_sunlit,
            double lai_o_shaded,
            double lai_u_sunlit,
            double lai_u_shaded,
            double clumping,
            double temp_air,
            double rh,
            double albedo_snow_v,
            double albedo_snow_n,
            double percentArea_snow_o,
            double percentArea_snow_u,
            double percent_snow_g,
            double albedo_v_o,
            double albedo_n_o,
            double albedo_v_u,
            double albedo_n_u,
            double albedo_v_g,
            double albedo_n_g,
            double * netRad_o,
            double * netRad_u,
            double * netRad_g,
            double * netRadLeaf_o_sunlit,
            double * netRadLeaf_o_shaded,
            double * netRadLeaf_u_sunlit,
            double * netRadLeaf_u_shaded,
            double * netShortRadLeaf_o_sunlit,
            double * netShortRadLeaf_o_shaded,
            double * netShortRadLeaf_u_sunlit,
            double * netShortRadLeaf_u_shaded )
```

Function to calculate net radiation at canopy level and leaf level.
[input] global solar radiation, cosine value for solar zenith angle, albedo of leaves albedo of snow, percentage of snow cover, leaf area index overstorey and understorey, temperature of overstorey, understorey and ground (contain snow?) temperature of air (C), relative humidity (0-100)
[output] net radiation for canopy, overstorey, understorey and ground; net radiation on sunlit, shaded leaves of overstorey and understorey.

Parameters

shortRad_global	global short radiation
CosZs	cosine value of solar zenith angle
temp_o	temperature of overstorey
temp_u	temperature of understory
temp_g	temperature of ground
lai_o	leaf area index of overstory, without stem
lai_u	leaf area index of understory, without stem
lai_os	leaf area index of overstory, with stem
lai_us	leaf area index of understory, with stem
lai_o_sunlit	sunlit leaves LAI with consideration of stem, overstory
lai_o_shaded	shaded leaves LAI with consideration of stem, overstory
lai_u_sunlit	sunlit leaves LAI with consideration of stem, understory
lai_u_shaded	shaded leaves LAI with consideration of stem, understory
clumping	clumping index
temp_air	air temperature
rh	relative humidity
albedo_snow_v	albedo of snow in this step, visible
albedo_snow_n	albedo of snow in this step, near infrared
percentArea_snow_o	percentage of snow on overstorey (by area)
percentArea_snow_u	percentage of snow on understorey (by area)
percent_snow_g	percentage of snow on ground (by mass)
albedo_v_o	albedo of overstory, visible, not considering snow, decided by land cover
albedo_n_o	albedo of overstory, near infrared
albedo_v_u	albedo of understory, visible
albedo_n_u	albedo of understory, near infrared
albedo_v_g	albedo of ground, visible
albedo_n_g	albedo of ground, near infrared
netRad_o	net radiation on overstorey
netRad_u	net radiation on understorey
netRad_g	net radiation on ground
netRadLeaf_o_sunlit	net radiation at the leaf level, overstory sunlit, for ET calculation
netRadLeaf_o_shaded	net radiation at the leaf level, overstory shaded
netRadLeaf_u_sunlit	net radiation at the leaf level, understory sunlit
netRadLeaf_u_shaded	net radiation at the leaf level, understory shaded
netShortRadLeaf_o_sunlit	net shortwave radiation at leaf level, overstory sunlit, for GPP calculation
netShortRadLeaf_o_shaded	net shortwave radiation at leaf level, overstory shaded
netShortRadLeaf_u_sunlit	net shortwave radiation at leaf level, understory sunlit
netShortRadLeaf_u_shaded	net shortwave radiation at leaf level, understory shaded

Returns

void

5.2.2.10 photosynthesis()

```
void photosynthesis (
            double temp_leaf_p,
            double rad_leaf,
            double e_air,
            double g_lb_w,
            double vc_opt,
            double f_soilwater,
            double b_h2o,
            double m_h2o,
            double cii,
            double temp_leaf_c,
            double LH_leaf,
            double * Gs_w,
            double * aphoto,
            double * ci )
```

Function to calculate leaf photosynthesis by solving a cubic equation.
[output] stomatal conductance to water vapor (m s-1); net photosynthesis rate (umol CO2 m-2 s-1); intercellular co2 concentration (ppm)

Parameters

temp_leaf_p	temporary variables, to be removed later
rad_leaf	net shortwave radiation (W/m2)
e_air	water vapor pressure above canopy (kPa)
g_lb_w	leaf laminar boundary layer conductance to H2O (m/s)
vc_opt	the maximum velocities of carboxylation of Rubisco at 25 deg C (umol m-2 s-1)
f_soilwater	an empirical scalar of soil water stress on stomatal conductance, dimensionless
b_h2o	the intercept term in BWB model (mol H2O m-2 s-1)
m_h2o	the slope in BWB model
cii	initial intercellular co2 concentration (ppm)
temp_leaf_c	leaf temperature (deg C)
LH_leaf	leaf latent heat flux (W m-2)
Gs_w	stomatal conductance to water vapor (m s-1)
aphoto	net photosynthesis rate (umol CO2 m-2 s-1)
ci	intercellular co2 concentration (ppm)

Returns

void

5.2.2.11 plantresp()

```
void plantresp (
            int LC,
            struct results * mid_res,
```

```
            double lai_yr,
            double lai,
            double temp_air,
            double temp_soil,
            double CosZs )
```

Function to calculate plant respiration.

Parameters

LC	land cover type
mid_res	results struct
lai_yr	annual mean leaf area index
lai	daily leaf area index
temp_air	air temperature
temp_soil	soil temperature
CosZs	cosine of solar zenith angle

Returns

void

5.2.2.12 rainfall_stage1()

```
void rainfall_stage1 (
            double temp_air,
            double precipitation,
            double mass_water_o_last,
            double mass_water_u_last,
            double lai_o,
            double lai_u,
            double clumping,
            double * mass_water_o,
            double * mass_water_u,
            double * percent_water_o,
            double * percent_water_u,
            double * precipitation_g )
```

Function of rainfall stage1.
[rainfall_stage1] happens before evaporation of intercepted water from canopy (supply)
[input] air temperature, precipitation (m/s), remain of water on leaves from last step (kg/m2) per leaf area leaf area index of overstorey and understorey, excluding stem. length of this step (s), if time step is 10min, then it is set as 600, air temperature and humidity
[output] percentage of canopy covered by rainfall, overstorey and understorey (provided to evaporation_canopy), mass of water available for evaporation on canopy in this step precipitation on ground
[optical output] intercepted mass of rainfall in this step

Parameters

temp_air	air temperature (Celsius)
precipitation	precipitation rate (m/s)
mass_water_o_last	remains of water from last step, overstory
mass_water_u_last	remains of water from last step, understory
lai_o	leaf area index, overstory
lai_u	leaf area index, understory
clumping	clumping index

Parameters

mass_water_o	mass of water on leaves (kg/m2) per ground area, overstory
mass_water_u	mass of water on leaves (kg/m2) per ground area, understory
percent_water_o	the fraction of canopy covered by liquid water and snow, overstory
percent_water_u	the fraction of canopy covered by liquid water and snow, understory
precipitation_g	precipitation on ground

Returns

void

5.2.2.13 rainfall_stage2()

```
void rainfall_stage2 (
            double evapo_water_o,
            double evapo_water_u,
            double * mass_water_o,
            double * mass_water_u )
```

Function of rainfall stage2.
[rainfall_stage2] happens after evaporation of intercepted water from canopy (demand)
[input] mass of water on leaves after precipitation in this step, evaporation from leaves in this step
[output] mass of water on leaves after the evaporation on leaves in this step (this value is transferred to next step)

Parameters

evapo_water↵_o	evaporation of intercepted rain in this step, overstorey, kg/m2/s = mm/s
evapo_water↵_u	evaporation of intercepted rain in this step, understorey, kg/m2/s = mm/s
mass_water↵_o	supply of rain on leaves, overstory, already added precipitation in this step
mass_water↵_u	supply of rain on leaves, understory, already added precipitation in this step

Returns

void

5.2.2.14 s_coszs()

```
void s_coszs (
            short jday,
            short j,
            float lat,
            float lon,
            double * CosZs )
```

Function to calculate cosine solar zenith angle.

Parameters

jday	date of year
j	local time/UTC time code needs to be edited according to time format

Parameters

lat	latitude of site
lon	longitude of site
CosZs	cosine solar zenith angle

Returns

void

5.2.2.15 sensible_heat()

```
void sensible_heat (
            double tempL_o_sunlit,
            double tempL_o_shaded,
            double tempL_u_sunlit,
            double tempL_u_shaded,
            double temp_g,
            double temp_air,
            double rh_air,
            double Gheat_o_sunlit,
            double Gheat_o_shaded,
            double Gheat_u_sunlit,
            double Gheat_u_shaded,
            double Gheat_g,
            double lai_o_sunlit,
            double lai_o_shaded,
            double lai_u_sunlit,
            double lai_u_shaded,
            double * SH_o,
            double * SH_u,
            double * SH_g )
```

Function to calculate sensible heat.
[input] temperature of sunlit and shaded leaves from other storey (leaf temperature module); temperature of air; relative humidity; temperature of ground (soil heat flux module); aerodynamic heat conductance of sunlit shaded leaves from overstorey and understorey; aerodynamic heat conductance of ground; leaf area index, sunlit and shaded, overstorey and understorey (from leaf area index module);
[output] sensible heat from overstorey, understorey and ground

Parameters

tempL_o_sunlit	temperature of leaves, overstory, sunlit
tempL_o_shaded	temperature of leaves, overstory, shaded
tempL_u_sunlit	temperature of leaves, understory, sunlit
tempL_u_shaded	temperature of leaves, understory, shded
temp_g	temperature of ground
temp_air	air temperature
rh_air	relative humidity of air
Gheat_o_sunlit	aerodynamic resistance of heat, overstory, sunlit
Gheat_o_shaded	aerodynamic resistance of heat, overstory, shaded
Gheat_u_sunlit	aerodynamic resistance of heat, understory, sunlit
Gheat_u_shaded	aerodynamic resistance of heat, understory, shaded
Gheat_g	aerodynamic resistance of heat, ground
lai_o_sunlit	leaf area index, overstory, sunlit

Parameters

lai_o_shaded	leaf area index, overstory, shaded
lai_u_sunlit	leaf area index, understory, sunlit
lai_u_shaded	leaf area index, understory, shaded
SH_o	sensible heat, overstory
SH_u	sensible heat, understory
SH_g	sensible heat, ground

Returns

void

5.2.2.16 snowpack_stage1()

```
void snowpack_stage1 (
            double temp_air,
            double precipitation,
            double mass_snow_o_last,
            double mass_snow_u_last,
            double mass_snow_g_last,
            double * mass_snow_o,
            double * mass_snow_u,
            double * mass_snow_g,
            double lai_o,
            double lai_u,
            double clumping,
            double * area_snow_o,
            double * area_snow_u,
            double * percent_snow_o,
            double * percent_snow_u,
            double * percent_snow_g,
            double * density_snow,
            double * depth_snow,
            double * albedo_v_snow,
            double * albedo_n_snow )
```

Function of snowpack stage1.
[snowpack_stage1] happens before any consumption of snow in this step, after the snow fall (supply)
[Input] air temperature, precipitation,depth of snow from last step, density of snow from last step, mass of snow on canopy and ground (per ground area) from last step, length of step, leaf area index of overstorey and understorey excluding stem, albedo of snow from last step.
[Output] mass of snow on canopy and ground accumulation of snowfall, albedo of snow in this step, density of snow in this step.

Parameters

temp_air	air temperature
precipitation	precipitation (m/s)
mass_snow_o_last	weight of snow at overstorey from last step
mass_snow_u_last	weight of snow at understorey from last step
mass_snow_g_last	weight of snow on ground from last step
mass_snow_o	mass of intercepted snow at overstory, input from last step, kg/m2
mass_snow_u	mass of intercepted snow at understory, input from last step, kg/m2
mass_snow_g	mass of intercepted snow on ground, input from last step, kg/m2

Parameters

lai_o	overstory lai
lai_u	understory lai
clumping	clumping index
area_snow_o	area of snow at overstorey
area_snow_u	area of snow at understorey
percent_snow_o	percentage of snow cover at overstory, DECIDED by weight
percent_snow_u	percentage of snow cover at understory, DECIDED by weight
percent_snow_g	percentage of snow cover on ground, DECIDED by weight
density_snow	density of snowpack on ground, input from last step, then changed in this module
depth_snow	depth of snowpack, input from last step, changed here, then changed in stage2
albedo_v_snow	visible albedo of snow, input from this step, changed in this module
albedo_n_snow	near infrared albedo of snow, input from this step, changed in this module

Returns

void

5.2.2.17 snowpack_stage2()

```
void snowpack_stage2 (
            double evapo_snow_o,
            double evapo_snow_u,
            double * mass_snow_o,
            double * mass_snow_u )
```

Function of snowpack stage2. This module will calculate the snow remained on canopy surface after evaporation in this step.
[snowpack_stage2] happens after sublimation from ground and canopy (demand)
[input] mass of snow on leaves after precipitation in this step, sublimation from leaves in this step
[output] mass of snow on leaves after the sublimation on leaves in this step

Parameters

evapo_snow←_o	evaporation of intercepted rain in this step, overstorey, kg/m2/s = mm/s
evapo_snow←_u	evaporation of intercepted rain in this step, understorey, kg/m2/s = mm/s
mass_snow←_o	supply of rain on leaves, overstorey, already added precipitation in this step
mass_snow←_u	supply of rain on leaves, understorey, already added precipitation in this step

Returns

void

5.2.2.18 snowpack_stage3()

```
void snowpack_stage3 (
            double temp_air,
            double temp_snow,
```

```
            double temp_snow_last,
            double density_snow,
            double * depth_snow,
            double * depth_water,
            double * mass_snow_g )
```

Function of snowpack stage3. This module simulates the process of snow melting and water frozen in this step.
[snowpack stage3] happens after frozen and melt of snow pack (demand)
[input] depth of snow on ground after stage 1, air temperature, ground surface temperature
[output] the amount of the melted snow, frozen snow

Parameters

temp_air	temperature of air in this step
temp_snow	temperature of snow in this step
temp_snow_last	temperature of snow in last step
density_snow	density of snow output from stage1
depth_snow	depth of snow on ground after stage1
mass_snow_g	mass of snow on ground after stage1
depth_water	depth of water after all precipitation and evaporation

Returns

void

5.2.2.19 soilresp()

```
void soilresp (
            double * Ccd,
            double * Cssd,
            double * Csmd,
            double * Cfsd,
            double * Cfmd,
            double * Csm,
            double * Cm,
            double * Cs,
            double * Cp,
            float npp_yr,
            double * coef,
            int soiltype,
            struct Soil * soilp,
            struct results * mid_res )
```

Function to calculate soil respiration.

Parameters

Ccd	carbon pool variable
Cssd	...
Csmd	...
Cfsd	...
Cfmd	...
Csm	...
Cm	...
Cs	...
Cp	...

Parameters

npp_yr	a fraction of NPP transferred to biomass carbon pools
coef	soil coefficients array
soiltype	soil type
soilp	soil variables struct
mid_res	results struct

Returns

void

NEP

5.2.2.20 surface_temperature()

```
void surface_temperature (
            double temp_air,
            double rh_air,
            double depth_snow,
            double depth_water,
            double capacity_heat_soil1,
            double capacity_heat_soil0,
            double Gheat_g,
            double depth_soil1,
            double density_snow,
            double tempL_u,
            double netRad_g,
            double evapo_soil,
            double evapo_water_g,
            double evapo_snow_g,
            double lambda_soil1,
            double percent_snow_g,
            double heat_flux_soil1,
            double temp_ground_last,
            double temp_soil1_last,
            double temp_any0_last,
            double temp_snow_last,
            double temp_soil0_last,
            double temp_snow1_last,
            double temp_snow2_last,
            double * temp_ground,
            double * temp_any0,
            double * temp_snow,
            double * temp_soil0,
            double * temp_snow1,
            double * temp_snow2,
            double * heat_flux )
```

Function to simulate surface temperature, and heat flux from surface to soil layers.

Parameters

temp_air	air temperature (Celsius degree)
rh_air	relative humidity (0-100)
depth_snow	depth of snow (m)
depth_water	depth of water on ground (m)

Parameters

capacity_heat_soil1	heat capacity of layer1 soil (J/m2/K)
capacity_heat_soil0	heat capacity of layer2 soil (J/m2/K)
Gheat_g	aerodynamic conductance of heat on ground (m/s)
depth_soil1	depth of soil in layer1 (m)
density_snow	density of snow (kg/m3)
tempL_u	leaf temperature, understory (Celsius degree)
netRad_g	net radiation on ground (W/m2)
evapo_soil	evaporation from soil surface (mm/s)
evapo_water_g	evaporation from pond water on ground (mm/s)
evapo_snow_g	evaporation from snow pack on ground (mm/s)
lambda_soil1	thermal conductivity of layer1 soil (W/m/K)
percent_snow_g	percentage of snow coverage on ground (0-1)
heat_flux_soil1	heat flux from layer1 soil to the next soil layer (W/m2)
temp_ground_last	temperature of ground, from last step
temp_soil1_last	temperature of layer1 soil, from last step
temp_any0_last	temperature of any layer right above the soil, from last step
temp_snow_last	temperature of snow, from last step
temp_soil0_last	temperature of soil0, from last step
temp_snow1_last	temperature of snow layer 2, from last step
temp_snow2_last	temperature of snow layer 3, from last step
temp_ground	ground temperature at this step
temp_any0	temperature of any layer right above the soil could be a mixture of snow temperature and soil surface temperature
temp_snow	snow temperature at this step
temp_soil0	temperature of soil surface right above the soil, the part not covered by snow
temp_snow1	temperature of snow layer 2, used in depth_snow>0.05 m
temp_snow2	temperature of snow layer 3, used in depth_snow>0.05 m
heat_flux	heat flux from ground to soil

Returns

void

5.2.2.21 transpiration()

```
void transpiration (
            double tempL_o_sunlit,
            double tempL_o_shaded,
            double tempL_u_sunlit,
            double tempL_u_shaded,
            double temp_air,
            double rh_air,
            double Gtrans_o_sunlit,
            double Gtrans_o_shaded,
            double Gtrans_u_sunlit,
            double Gtrans_u_shaded,
            double lai_o_sunlit,
            double lai_o_shaded,
            double lai_u_sunlit,
```

```
              double lai_u_shaded,
              double * trans_o,
              double * trans_u )
```

Function to calculate transpiration.
A transformation of Penman-Monteith equation is used here. It could be regarded as a mass transfer process. Water vapor inside cells are required by VPD from air and VPD on leaf surface.
[input] temperature of sunlit and shaded leaves from other storey (leaf temperature module); temperature of air; relative humidity; conductance of water for sunlit shaded leaves from overstorey and understorey; leaf area index, sunlit and shaded, overstorey and understorey (from leaf area index module);
[output] transpiration from overstorey and understorey

Parameters

tempL_o_sunlit	temperature of leaf, overstory, sunlit
tempL_o_shaded	temperature of leaf, overstory, shaded
tempL_u_sunlit	temperature of leaf, understory, sunlit
tempL_u_shaded	temperature of leaf, understory, shaded
temp_air	air temperature
rh_air	relative humidity of air
Gtrans_o_sunlit	total conductance of water tandem of stomatal conductance and aerodynamic conductance, overstory, sunlit
Gtrans_o_shaded	..., overstory, shaded
Gtrans_u_sunlit	..., understory, sunlit
Gtrans_u_shaded	..., understory, shaded
lai_o_sunlit	leaf area index, overstory, sunlit
lai_o_shaded	leaf area index, overstory, shaded
lai_u_sunlit	leaf area index, understory, sunlit
lai_u_shaded	leaf area index, understory, shaded
trans_o	transpiration from overstory
trans_u	transpiration from understory

Returns

void

5.2.2.22 Vcmax_Jmax()

```
void Vcmax_Jmax (
              double lai_o,
              double clumping,
              double Vcmax0,
              double slope_Vcmax_N,
              double leaf_N,
              double CosZs,
              double * Vcmax_sunlit,
              double * Vcmax_shaded,
              double * Jmax_sunlit,
              double * Jmax_shaded )
```

Function to calculate the Vcmax and Jmax for sunlit and shaded leaf.

Note

Vcmax0 is for the leaf vcmax at top of the canopy. LHE

Just to clarify, in this version, Vcmax0 is still the average leaf Vcmax25, Vcmax0 $*$ slope_Vcmax_N $*$ leaf_N is the top leaves Vcmax25. XL. 20190403.

Parameters

lai_o	overstory lai
clumping	clumping index
Vcmax0	maximum capacity of Rubisco at 25C-Vcmax
slope_Vcmax←_N	slope of Vcmax-N curve
leaf_N	leaf Nitrogen content mean value + 1 SD g/m2
CosZs	cosine solar zenith angle
Vcmax_sunlit	Vcmax of sunlit leaf
Vcmax_shaded	Vcmax of shaded leaf
Jmax_sunlit	Jmax of sunlit leaf
Jmax_shaded	Jmax of shaded leaf

Returns

void

5.3 beps.h

Go to the documentation of this file.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include "soil.h"

#define NOERROR     0
#define ERROR       1
#define PI          3.1415926
#define zero        0.0000000001

//#define max(a,b)  (a>b)?a:b         // used for UNIX
//#define min(a,b)  (a<b)?a:b         // used for UNIX
#define max(a,b)    ((a)>(b))?(a):(b) // LHE. the  original one can lead to disorder.
#define min(a,b)    ((a)<(b))?(a):(b) // LHE


#define l_sta       105    // start line
#define l_end       105    // end line
#define p_sta       101    // start pix
#define p_end       101    // end pix

#define RTIMES      24      // 24
#define step        3600    // 3600 in sec
#define kstep       360     // 10 times per hour, 360 sec. per time
#define kloop       10      // 10 times per hour, 360 sec. per time
#define layer       5
#define depth_f     6
#define CO2_air     380     // atmospheric CO2 concentration
#define rho_a       1.292   // density of air at 0C

struct climatedata
{
    double Srad;
    double LR;
    double temp;
    double rh;
    double rain;
    double wind;
    double dr_o;
    double df_o;
    double dr_u;
    double df_u;
//  float st_c;
};

struct results
{
    double gpp_o_sunlit;
    double gpp_u_sunlit;
```

```
    double gpp_o_shaded;
    double gpp_u_shaded;
    double plant_resp;
    double npp_o;
    double npp_u;
    double GPP;
    double NPP;
    double NEP;
    double soil_resp;
    double Net_Rad;
    double SH;
    double LH;
    double Trans;
    double Evap;
};

struct cpools
{
    double Ccd[3];
    double Cssd[3];
    double Csmd[3];
    double Cfsd[3];
    double Cfmd[3];
    double Csm[3];
    double Cm[3];
    double Cs[3];
    double Cp[3];
};

void readconf();
void mid_prg();
void readinput1();
void readlai_d();
void readlonlat();

void inter_prg(int jday,int rstep,double lai,double clumping,double parameter[],struct climatedata* meteo,
                double CosZs,double var_o[],double var_n[],struct Soil* soilp,struct results* mid_res);

void s_coszs(short jday,short j,float lat,float lon,double* CosZs);

void aerodynamic_conductance(double canopy_height_o, double canopy_height_u, double zz, double clumping,
                             double temp_air, double wind_sp, double SH_o_p, double lai_o, double lai_u,
                             double *rm, double *ra_u, double *ra_g, double *G_o_a, double *G_o_b, double *G_u_a, double *G_u_b);
void plantresp(int LC, struct results* mid_res, double lai_yr, double lai,double temp_air, double temp_soil, double CosZs);

void Vcmax_Jmax(double lai_o, double clumping, double Vcmax0,
                double slope_Vcmax_N, double leaf_N, double CosZs,
                double *Vcmax_sunlit, double *Vcmax_shaded, double *Jmax_sunlit, double *Jmax_shaded);

void netRadiation(double shortRad_global, double CosZs, double temp_o, double temp_u, double temp_g,
                  double lai_o, double lai_u, double lai_os, double lai_us,
                  double lai_o_sunlit, double lai_o_shaded, double lai_u_sunlit, double lai_u_shaded,
                  double clumping, double temp_air, double rh,
                  double albedo_snow_v, double albedo_snow_n, double percentArea_snow_o, double percentArea_snow_u, double percent_snow_g,
                  double albedo_v_o, double albedo_n_o, double albedo_v_u, double albedo_n_u, double albedo_v_g, double albedo_n_g,
                  double* netRad_o, double* netRad_u, double* netRad_g,
                  double* netRadLeaf_o_sunlit, double* netRadLeaf_o_shaded, double* netRadLeaf_u_sunlit, double* netRadLeaf_u_shaded,
                  double* netShortRadLeaf_o_sunlit, double* netShortRadLeaf_o_shaded, double* netShortRadLeaf_u_sunlit, double* netShortRadLeaf_u_shaded);

void soilresp(double* Ccd, double* Cssd, double* Csmd, double* Cfsd, double* Cfmd,
              double* Csm, double* Cm, double* Cs, double* Cp, float npp_yr, double* coef,
              int soiltype, struct Soil* soilp,struct results* mid_res);

void readparam(short lc, double parameter1[]);

void lai2(double stem_o,double stem_u,int LC,double CosZs,double lai_o,double clumping,double lai_u,
          double* lai_o_sunlit,double* lai_o_shaded,double* lai_u_sunlit,double* lai_u_shaded,
          double* PAI_o_sunlit,double* PAI_o_shaded,double* PAI_u_sunlit,double* PAI_u_shaded);

void readcoef(short lc, int stxt, double coef[]);

void readhydr_param();

void photosynthesis(double temp_leaf_p,double rad_leaf, double e_air, double g_lb_w, double vc_opt,
                    double f_soilwater,double b_h2o, double m_h2o, double cii,double temp_leaf_c,double LH_leaf,
                    double* Gs_w, double* aphoto, double* ci);
void soil_water_factor();
void Leaf_Temperatures(double Tair, double slope, double psychrometer, double VPD_air, double Cp_ca,
                       double Gw_o_sunlit, double Gw_o_shaded, double Gw_u_sunlit, double Gw_u_shaded,
```

```
                          double Gww_o_sunlit, double Gww_o_shaded, double Gww_u_sunlit, double Gww_u_shaded,
                          double Gh_o_sunlit, double Gh_o_shaded, double Gh_u_sunlit, double Gh_u_shaded,
                          double Xcs_o, double Xcl_o, double Xcs_u, double Xcl_u,
                          double radiation_o_sun, double radiation_o_shaded, double radiation_u_sun, double radiation_u_shaded,
                          double *Tc_o_sunlit, double *Tc_o_shaded, double *Tc_u_sunlit, double *Tc_u_shaded);

double Leaf_Temperature(double Tair, double slope, double psychrometer, double VPD_air, double Cp_ca,
                           double Gw, double Gww, double Gh, double Xcs, double Xcl, double radiation);

void sensible_heat(double tempL_o_sunlit, double tempL_o_shaded, double tempL_u_sunlit, double tempL_u_shaded,
                   double temp_g, double temp_air, double rh_air,
                   double Gheat_o_sunlit, double Gheat_o_shaded, double Gheat_u_sunlit, double Gheat_u_shaded, double Gheat_g,
                   double lai_o_sunlit, double lai_o_shaded, double lai_u_sunlit, double lai_u_shaded,
                   double* SH_o, double* SH_u, double* SH_g);

void transpiration(double tempL_o_sunlit, double tempL_o_shaded, double tempL_u_sunlit, double tempL_u_shaded,
                   double temp_air, double rh_air,
                   double Gtrans_o_sunlit, double Gtrans_o_shaded, double Gtrans_u_sunlit, double Gtrans_u_shaded,
                   double lai_o_sunlit, double lai_o_shaded, double lai_u_sunlit, double lai_u_shaded,
                   double* trans_o, double* trans_u);

void evaporation_canopy(double tempL_o_sunlit, double tempL_o_shaded, double tempL_u_sunlit, double tempL_u_shaded,
                        double temp_air, double rh_air,
                        double Gwater_o_sunlit, double Gwater_o_shaded, double Gwater_u_sunlit, double Gwater_u_shaded,
                        double lai_o_sunlit, double lai_o_shaded, double lai_u_sunlit, double lai_u_shaded,
                        double percent_water_o, double percent_water_u, double percent_snow_o, double percent_snow_u,
                        double* evapo_water_o, double* evapo_water_u, double* evapo_snow_o, double* evapo_snow_u);

void evaporation_soil(double temp_air, double temp_g, double rh_air, double netRad_g, double Gheat_g,
                      double* percent_snow_g,double* depth_water, double* depth_snow, double* mass_water_g, double* mass_snow_g,
                      double density_snow, double swc_g, double porosity_g,
                      double* evapo_soil, double* evapo_water_g, double* evapo_snow_g);

void rainfall_stage1(double temp_air, double precipitation, double mass_water_o_last, double mass_water_u_last,
                     double  lai_o, double lai_u, double clumping,
                     double* mass_water_o, double* mass_water_u,
                     double* percent_water_o, double* percent_water_u, double* precipitation_g);

void rainfall_stage2(double evapo_water_o, double evapo_water_u,
                     double* mass_water_o, double* mass_water_u);
void rainfall_stage3();

void meteo_pack(double temp, double rh, double* meteo_pack_output);

void surface_temperature(double temp_air,double rh_air, double depth_snow, double depth_water,
                         double capacity_heat_soil1, double capacity_heat_soil0, double Gheat_g,
                         double depth_soil1, double density_snow,double tempL_u, double netRad_g,
                         double evapo_soil, double evapo_water_g, double evapo_snow_g, double lambda_soil1,
                         double percent_snow_g, double heat_flux_soil1, double temp_ground_last,
                         double temp_soil1_last, double temp_any0_last, double temp_snow_last,
                         double temp_soil0_last, double temp_snow1_last, double temp_snow2_last,
                         double* temp_ground, double* temp_any0, double* temp_snow,
                         double* temp_soil0, double* temp_snow1, double* temp_snow2, double* heat_flux);

void snowpack_stage1(double temp_air, double precipitation,double mass_snow_o_last, double mass_snow_u_last,
                     double mass_snow_g_last, double* mass_snow_o, double* mass_snow_u, double* mass_snow_g,
                     double lai_o, double lai_u, double clumping, double* area_snow_o, double* area_snow_u,
                     double* percent_snow_o, double* percent_snow_u, double* percent_snow_g,
                     double* density_snow, double* depth_snow, double* albedo_v_snow, double* albedo_n_snow);

void snowpack_stage2(double evapo_snow_o, double evapo_snow_u,
                     double* mass_snow_o, double* mass_snow_u);

void snowpack_stage3(double temp_air, double temp_snow, double temp_snow_last, double density_snow,
                     double* depth_snow, double* depth_water, double* mass_snow_g);

short lc_no;
int yr,bgn_day,end_day;
```

```
int npixels,nlines;
char lc_fn[255];         /* Land cover file */
char lai_fn[255];        /* Leaf area index file  */
char lai_fp[255];        /* Leaf area index file prefix */
char stxt_fn[255];       /* soil texture file */
char ci_fn[255];         /* clumping index file */
char st_fn[255];         /* initial values of soil temp */
char sw_fn[255];         /* initial values of soil water */
char sdp_fn[255];        /* initial values of snow depth*/
char r_fn[255];          /* meteor. data files */
char t_fn[255];
char h_fn[255];
char p_fn[255];
char wd_fn[255];
char lon_fn[255];
char lat_fn[255];
char fp4outp1[255];       /* output file1 prefix */
char fp4outp2[255];       /* output file2 prefix */
char fp4outp3[255];       /* output file3 prefix */
```

5.4 bepsmain_pnt.c File Reference

Main function. BEPS, for Boreal Ecosystems Productivity Simulator. BEPS 4.01 for a point, to simulate carbon fluxes, energy fluxes and soil water...

```
#include "beps.h"
#include "soil.h"
```

Functions

- int main ()
  Main driver function of BEPS.

5.4.1 Detailed Description

Main function. BEPS, for Boreal Ecosystems Productivity Simulator. BEPS 4.01 for a point, to simulate carbon fluxes, energy fluxes and soil water...

5.4.2 Function Documentation

5.4.2.1 main()

```
int main ( )
```

Main driver function of BEPS.

Read daily lai and hourly meteor. data

5.5 calc_temp_leaf.c File Reference

Subroutine to calculate the sunlit and shaded leaf temperatures for overstory and understory leave.

```
#include "beps.h"
```

Functions

- void Leaf_Temperatures (double Tair, double slope, double psychrometer, double VPD_air, double Cp_ca, double Gw_o_sunlit, double Gw_o_shaded, double Gw_u_sunlit, double Gw_u_shaded, double Gww_o_sunlit, double Gww_o_shaded, double Gww_u_sunlit, double Gww_u_shaded, double Gh_o_sunlit, double

Gh_o_shaded, double Gh_u_sunlit, double Gh_u_shaded, double Xcs_o, double Xcl_o, double Xcs_u, double Xcl_u, double radiation_o_sun, double radiation_o_shaded, double radiation_u_sun, double radiation_↩u_shaded, double ∗Tc_o_sunlit, double ∗Tc_o_shaded, double ∗Tc_u_sunlit, double ∗Tc_u_shaded)

Function to calculate leaf temperature four components (sunlit and shaded leaves, overstory and understory)

- double Leaf_Temperature (double Tair, double slope, double psychrometer, double VPD_air, double Cp_ca, double Gw, double Gww, double Gh, double Xcs, double Xcl, double radiation)

  Subroutine to calculate leaf temperature.

5.5.1 Detailed Description

Subroutine to calculate the sunlit and shaded leaf temperatures for overstory and understory leave.

Authors

Written and refactored by Liming He (liming.he@gmail.com)

Original contributor: Weimin Ju

Date

Last update: Sept. 15, 2015

Created on May 15, 2015

5.5.2 Function Documentation

5.5.2.1 Leaf_Temperature()

```
double Leaf_Temperature (
            double Tair,
            double slope,
            double psychrometer,
            double VPD_air,
            double Cp_ca,
            double Gw,
            double Gww,
            double Gh,
            double Xcs,
            double Xcl,
            double radiation )
```

Subroutine to calculate leaf temperature.

Parameters

Tair	air temperature
slope	the slope of saturation vapor pressure-temperature curve
psychrometer	psychrometer constant, 0.066 kPa K
VPD_air	vapor pressure deficit
Cp_ca	specific heat of moist air in kJ/kg/K
Gw	total conductance for water from the intercellular space of the leaves to the reference height above the canopy
Gww	total conductance for water from the surface of the leaves to the reference height above the canopy
Gh	total conductance for heat transfer from the leaf surface to the reference height above the canopy
Xcs	the fraction of canopy covered by snow
Xcl	the fraction of canopy covered by liquid water
radiation	net radiation on leaves

Returns

[double Tc] the effective canopy temperature in Kalvin

5.5.2.2 Leaf_Temperatures()

```
void Leaf_Temperatures (
            double Tair,
            double slope,
            double psychrometer,
            double VPD_air,
            double Cp_ca,
            double Gw_o_sunlit,
            double Gw_o_shaded,
            double Gw_u_sunlit,
            double Gw_u_shaded,
            double Gww_o_sunlit,
            double Gww_o_shaded,
            double Gww_u_sunlit,
            double Gww_u_shaded,
            double Gh_o_sunlit,
            double Gh_o_shaded,
            double Gh_u_sunlit,
            double Gh_u_shaded,
            double Xcs_o,
            double Xcl_o,
            double Xcs_u,
            double Xcl_u,
            double radiation_o_sun,
            double radiation_o_shaded,
            double radiation_u_sun,
            double radiation_u_shaded,
            double * Tc_o_sunlit,
            double * Tc_o_shaded,
            double * Tc_u_sunlit,
            double * Tc_u_shaded )
```

Function to calculate leaf temperature four components (sunlit and shaded leaves, overstory and understory) [output] Tc_o_sunlit,Tc_o_shaded,Tc_u_sunlit,Tc_u_shaded

Parameters

Tair	air temperature
slope	the slope of saturation vapor pressure-temperature curve
psychrometer	psychrometer constant, 0.066 kPa K
VPD_air	vapor pressure deficit
Cp_ca	specific heat of moist air in kJ/kg/K
Gw_o_sunlit	total conductance for water from the intercellular space of the leaves to the reference height above the canopy, overstory, sunlit
Gw_o_shaded	..., overstory, shaded
Gw_u_sunlit	..., understory, sunlit
Gw_u_shaded	..., understory, shaded
Gww_o_sunlit	total conductance for water from the surface of the leaves to the reference height above the canopy, overstory, sunlit
Gww_o_shaded	..., overstory, shaded
Gww_u_sunlit	..., understory, sunlit

Parameters

Gww_u_shaded	..., understory, shaded
Gh_o_sunlit	total conductance for heat transfer from the leaf surface to the reference height above the canopy, overstory, sunlit
Gh_o_shaded	..., overstory, shaded
Gh_u_sunlit	..., understory, sunlit
Gh_u_shaded	..., understory, shaded
Xcs_o	the fraction of canopy covered by snow, overstory
Xcl_o	the fraction of canopy covered by liquid water, overstory
Xcs_u	the fraction of canopy covered by snow, understory
Xcl_u	the fraction of canopy covered by liquid water, understory
radiation_o_sun	net radiation on leaves, overstory, sunlit
radiation_o_shaded	net radiation on leaves, overstory, shaded
radiation_u_sun	net radiation on leaves, understory, sunlit
radiation_u_shaded	net radiation on leaves, understory, shaded
Tc_o_sunlit	the effective canopy temperature in Kalvin, overstory, sunlit
Tc_o_shaded	the effective canopy temperature in Kalvin, overstory, shaded
Tc_u_sunlit	the effective canopy temperature in Kalvin, understory, sunlit
Tc_u_shaded	the effective canopy temperature in Kalvin, understory, shaded

Returns

void

5.6 DB.h

```

#define PI180 0.017453292                    // pi divided by 180, radians per degree
#define PI9 2.864788976
#define PI2 6.283185307                      // 2 time pi

 struct meteorology {

                double ustar;                // friction velocity, m s-1
                double ustarnew;             // updated friction velocity with new H, m s-1
                 double rhova_g;              // absolute humidity, g m-3
                 double rhova_kg;             // absolute humidity, kg m-3
                 double sensible_heat_flux;   // sensible heat flux, W M-2
                 double H_old;                // old sensible heat flux, W m-2
                 double air_density;          // air density, kg m-3
                 double T_Kelvin;             // absolute air temperature, K
                 double press_kpa;            // station pressure, kPa
                 double press_bars;           // station pressure, bars
                 double press_Pa;             // pressure, Pa
                 double pstat273;             // gas constant computations
                 double air_density_mole;     // air density, mole m-3
                 double relative_humidity;    // relative humidity, ea/es(T)
                 double vpd;                  // vapor pressure deficit
                 double ir_in;                // infrared flux density
                 } met;

 // structure for plant and physical factors

                 struct factors {
                 double latent;      // latent heat of vaporization, J kg-1
                 double latent18;    // latent heat of vaporization times molecular mass of vapor, 18 g
      mol-1
                 double heatcoef;    // factor for sensible heat flux density
                 double a_filt;          // filter coefficients
                 double b_filt;      // filter coefficients
                 double co2;         // CO2 factor, ma/mc * rhoa (mole m-3)

         } fact;

 struct boundary_layer_resistances{

                 double vapor;                // resistance for water vapor, s/m
```

```
                double heat;                    // resistance for heat, s/m
                double co2;                     // resistance for CO2, s/m
                } bound_layer_res;

void   TBOLTZdouble();
double TEMP_FUNC();
double TBOLTZ();
void   photosynthesis();
double SFC_VPD();
double ES();
double LAMBDA();





#define  rugc   8.314              // J mole-1 K-1
//#define  vcopt  73.0   // carboxylation rate at optimal temperature, umol m-2 s-1
//#define  jmopt  170.0  // electron transport rate at optimal temperature, umol m-2 s-1
#define  rd25   0.34     // dark respiration at 25 C, rd25= 0.34 umol m-2 s-1
#define  pi4    12.5663706
        //  Universal gas constant

#define  rgc1000 8314            // gas constant times 1000.

        // Consts for Photosynthesis model and kinetic equations.
        // for Vcmax and Jmax.  Taken from Harley and Baldocchi (1995, PCE)
#define hkin  200000.0   // enthalpy term, J mol-1
#define skin  710.0      // entropy term, J K-1 mol-1
#define ejm   55000.0      // activation energy for electron transport, J mol-1
#define evc   55000.0      // activation energy for carboxylation, J mol-1

        // Enzyme constants & partial pressure of O2 and CO2
        // Michaelis-Menten K values. From survey of literature.

#define kc25   274.6   // kinetic coef for CO2 at 25 C, microbars
#define ko25   419.8   // kinetic coef for O2 at 25C,  millibars
#define o2     210.0   // oxygen concentration  mmol mol-1


        // tau is computed on the basis of the Specificity factor (102.33)
        // times Kco2/Kh2o (28.38) to convert for value in solution
        // to that based in air/
        // The old value was 2321.1.

        // New value for Quercus robor from Balaguer et al. 1996
        // Similar number from Dreyer et al. 2001, Tree Physiol, tau= 2710

#define  tau25  2904.12    //  tau coefficient
        //  Arrhenius constants
        //  Eact for Michaelis-Menten const. for KC, KO and dark respiration
        //  These values are from Harley
#define  ekc   80500.0     // Activation energy for K of CO2; J mol-1
#define  eko   14500.0     // Activation energy for K of O2, J mol-1
#define  erd   38000.0     // activation energy for dark respiration, eg Q10=2
#define  ektau -29000.0  // J mol-1 (Jordan and Ogren, 1984)
#define tk_25  298.16     // absolute temperature at 25 C
#define toptvc 301.0    // optimum temperature for maximum carboxylation
#define toptjm 301.0    // optimum temperature for maximum electron transport
#define eabole 45162      // activation energy for bole respiration for Q10 = 2.02


        // Constants for leaf energy balance

#define  sigma   5.67e-08   // Stefan-Boltzmann constant W M-2 K-4
#define  cp      1005.         // Specific heat of air, J KG-1 K-1
#define mass_air 29.0     // Molecular weight of air, g mole-1
#define mass_CO2 44.0       // molecular weight of CO2, g mole-1
#define dldt     -2370.0      // Derivative of the latent heat of vaporization

#define ep        0.98                      // emissivity of leaves
#define epm1      0.02                     // 1- ep
#define epsoil    0.98                // Emissivity of soil
#define epsigma   5.5566e-8            // ep*sigma
#define epsigma2  11.1132e-8       // 2*ep*sigma
#define epsigma4  22.2264e-8       //  4.0 * ep * sigma
#define epsigma6  33.3396e-8       //  6.0 * ep * sigma
#define epsigma8  44.448e-8        //  8.0 * ep * sigma
#define epsigma12 66.6792e-8       // 12.0 * ep * sigma
#define betfact   1.5                 // multiplication factor for aerodynamic
                                                  // sheltering, based on work by Grace and Wilson
        //  constants for the polynomial equation for saturation vapor pressure-T function, es=f(t)
#define a1en      617.4
#define a2en      42.22
#define a3en      1.675
#define a4en      0.01408
#define a5en      0.0005818
```

```
        // Minimum stomatal resistance, s m-1.
#define rsm       145.0
#define brs       60.0      // curvature coeffient for light response

        //   leaf quantum yield, electrons
#define  qalpha   0.22
#define  qalpha2  0.0484   // qalpha squared, qalpha2 = pow(qalpha, 2.0);

        // Leaf dimension. geometric mean of length and width (m)
#define  lleaf    0.1       // leaf length, m


        // Diffusivity values for 273 K and 1013 mb (STP) using values from Massman (1998) Atmos Environment
        // These values are for diffusion in air.  When used these values must be adjusted for
        // temperature and pressure
        // nu, Molecular viscosity


#define  nuvisc  13.27        // mm2 s-1
#define  nnu     0.00001327  // m2 s-1

        // Diffusivity of CO2

#define  dc    13.81         // mm2 s-1
#define  ddc   0.00001381    // m2 s-1

        //   Diffusivity of heat

#define  dh    18.69;         // mm2 s-1
#define  ddh   0.00001869    // m2 s-1


        //  Diffusivity of water vapor

#define  dv   21.78        // mm2 s-1
#define  ddv   0.00002178    // m2 s-1
```

5.7 debug.h

```
#ifdef DEBUG
#undef DEBUG
#endif
//#define DEBUG 1
#define DEBUG 0
```

5.8 evaporation_canopy.c File Reference

This module calculates evaporation and sublimation from canopy, from overstorey understorey sunlit and shaded.

```
#include "beps.h"
```

Functions

- void evaporation_canopy (double tempL_o_sunlit, double tempL_o_shaded, double tempL_u_sunlit, double tempL_u_shaded, double temp_air, double rh_air, double Gwater_o_sunlit, double Gwater_o_shaded, double Gwater_u_sunlit, double Gwater_u_shaded, double lai_o_sunlit, double lai_o_shaded, double lai_u_sunlit, double lai_u_shaded, double percent_water_o, double percent_water_u, double percent_snow_o, double percent_snow_u, double *evapo_water_o, double *evapo_water_u, double *evapo_snow_o, double *evapo_snow_u)

  Function to calculate evaporation and sublimation from canopy.

5.8.1 Detailed Description

This module calculates evaporation and sublimation from canopy, from overstorey understorey sunlit and shaded.

Author

Edited by XZ Luo

Date

May 25, 2015

5.8.2 Function Documentation

5.8.2.1 evaporation_canopy()

```
void evaporation_canopy (
            double tempL_o_sunlit,
            double tempL_o_shaded,
            double tempL_u_sunlit,
            double tempL_u_shaded,
            double temp_air,
            double rh_air,
            double Gwater_o_sunlit,
            double Gwater_o_shaded,
            double Gwater_u_sunlit,
            double Gwater_u_shaded,
            double lai_o_sunlit,
            double lai_o_shaded,
            double lai_u_sunlit,
            double lai_u_shaded,
            double percent_water_o,
            double percent_water_u,
            double percent_snow_o,
            double percent_snow_u,
            double * evapo_water_o,
            double * evapo_water_u,
            double * evapo_snow_o,
            double * evapo_snow_u )
```

Function to calculate evaporation and sublimation from canopy.
[input] temperature of sunlit and shaded leaves from other storey (leaf temperature module); temperature of air; relative humidity; aerodynamic conductance of water (snow) for sunlit shaded leaves from overstorey and understorey; percentage of overstorey or understorey covered by water or snow; leaf area index, sunlit and shaded, overstorey and understorey (from leaf area index module);
[output] evaporation of water and snow from overstorey and understorey

Parameters

tempL_o_sunlit	temperature of leaves, overstory, sunlit (leaf temperature module)
tempL_o_shaded	temperature of leaves, overstory, shaded
tempL_u_sunlit	temperature of leaves, understory, sunlit
tempL_u_shaded	temperature of leaves, understory, shaded
temp_air	air temperature
rh_air	relative humidity
Gwater_o_sunlit	aerodynamic conductance of water (snow) for overstory, sunlit leaves
Gwater_o_shaded	aerodynamic conductance of water (snow) for overstory, shaded leaves
Gwater_u_sunlit	aerodynamic conductance of water (snow) for understory, sunlit leaves
Gwater_u_shaded	aerodynamic conductance of water (snow) for understory, shaded leaves
lai_o_sunlit	leaf area index, overstory, sunlit (from leaf area index module)

Parameters

lai_o_shaded	leaf area index, overstory, shaded
lai_u_sunlit	leaf area index, understory, sunlit
lai_u_shaded	leaf area index, understory, shaded
percent_water_o	percentage of overstorey covered by water
percent_water_u	percentage of understorey covered by water
percent_snow_o	percentage of overstorey covered by snow
percent_snow_u	percentage of understorey covered by snow
evapo_water_o	evaporation of water from overstorey
evapo_water_u	evaporation of water from understorey
evapo_snow_o	evaporation of snow from overstorey
evapo_snow_u	evaporation of snow from understorey

Returns

void

5.9 evaporation_soil.c File Reference

This module will calculate evaporation from ground surface/top soil, and the evaporation of snow and pond water on surface.

`#include "beps.h"`

Functions

- void evaporation_soil (double temp_air, double temp_g, double rh_air, double netRad_g, double Gheat←_g, double ∗percent_snow_g, double ∗depth_water, double ∗depth_snow, double ∗mass_water_g, double ∗mass_snow_g, double density_snow, double swc_g, double porosity_g, double ∗evapo_soil, double ∗evapo_water_g, double ∗evapo_snow_g)

  Function to calculate evaporation from ground surface/top soil, and the evaporation of snow and pond water on surface.

5.9.1 Detailed Description

This module will calculate evaporation from ground surface/top soil, and the evaporation of snow and pond water on surface.

Author

Edited by XZ Luo

Date

May 25, 2015

5.9.2 Function Documentation

5.9.2.1 evaporation_soil()

```
void evaporation_soil (
            double temp_air,
            double temp_g,
```

```
double rh_air,
double netRad_g,
double Gheat_g,
double * percent_snow_g,
double * depth_water,
double * depth_snow,
double * mass_water_g,
double * mass_snow_g,
double density_snow,
double swc_g,
double porosity_g,
double * evapo_soil,
double * evapo_water_g,
double * evapo_snow_g )
```

Function to calculate evaporation from ground surface/top soil, and the evaporation of snow and pond water on surface.
[input] air temperature; ground surface temperature; relative humidity of ground (BEPS takes it as the air RH); percentage of snow cover on ground; depth of water; depth of snow soil water content on first soil layer; porosity of first soil layer
[output] evaporation from soil surface; depth of water and snow on ground after evaporation and sublimation

Parameters

temp_air	air temperature
temp_g	ground temperature
rh_air	relative humidity of air
netRad_g	net radiation on ground
Gheat_g	aerodynamic conductance of heat on ground surface
percent_snow_g	percentage of snow on ground
depth_water	depth of water on ground, after rainfall and snowfall stage 1, before evaporation. output after subtracting evaporation
depth_snow	depth of snow on ground, ...
mass_water_g	mass of water on ground, output after subtracting evaporation
mass_snow_g	mass of snow on ground, ...
density_snow	density of snow, from snowpack stage1
swc_g	soil water content (from last step)
porosity_g	porosity on ground
evapo_soil	evaporation from soil
evapo_water_g	evaporation from pond water
evapo_snow_g	evaporation from snow on surface

Returns

void

5.10 init_soil.c File Reference

Module for soil parameters and status initialization.

```
#include "soil.h"
#include <math.h>
```

Functions

- void Init_Soil_Parameters (int landcover, int stxt, double r_root_decay, struct Soil p[ ])

  Function to initialize soil parameters.
- void Init_Soil_Status (struct Soil p[ ], double Tsoil, double Tair, double Ms, double snowdepth)

  Function to initialize the soil status: soil temperature and moisture for each layer, ponded water, snow depth, et al.
- void SoilRootFraction (struct Soil soil[ ])

  Function to calculate the fraction of root in the soil for each soil layer.

5.10.1 Detailed Description

Module for soil parameters and status initialization.

Author

Liming He

Date

June 2, 2015

5.10.2 Function Documentation

5.10.2.1 Init_Soil_Parameters()

```
void Init_Soil_Parameters (
            int landcover,
            int stxt,
            double r_root_decay,
            struct Soil p[ ] )
```

Function to initialize soil parameters.
[1] Set the depth for each layer
[2] Set the parameters for each layer

Parameters

landcover	land cover type
stxt	soil texture
r_root_decay	decay rate of root distribution
p	Soil struct variable

Returns

void

5.10.2.2 Init_Soil_Status()

```
void Init_Soil_Status (
            struct Soil p[ ],
            double Tsoil,
            double Tair,
            double Ms,
            double snowdepth )
```

Function to initialize the soil status: soil temperature and moisture for each layer, ponded water, snow depth, et al.

Parameters

p	Soil struct variable
Tsoil	soil temperature
Tair	air temperature
Ms	soil water content
snowdepth	snow depth

Returns

void

5.10.2.3 SoilRootFraction()

```
void SoilRootFraction (
            struct Soil soil[ ] )
```

Function to calculate the fraction of root in the soil for each soil layer.
Declare functions.

Parameters

soil	Soil struct variable

Returns

void

5.11 inter_prg.c File Reference

the inter-program between main program and modules

```
#include "beps.h"
#include "soil.h"
```

Functions

- void inter_prg (int jday, int rstep, double lai, double clumping, double parameter[ ], struct climatedata ∗meteo, double CosZs, double var_o[ ], double var_n[ ], struct Soil ∗soilp, struct results ∗mid_res)

    the inter-module function between main program and modules

5.11.1 Detailed Description

the inter-program between main program and modules

Date

Last update: July, 2015

5.11.2 Function Documentation

5.11.2.1 inter_prg()

```
void inter_prg (
            int jday,
```

```
int rstep,
double lai,
double clumping,
double parameter[ ],
struct climatedata * meteo,
double CosZs,
double var_o[ ],
double var_n[ ],
struct Soil * soilp,
struct results * mid_res )
```

the inter-module function between main program and modules

Parameters

jday	day of year
rstep	hour of day
lai	leaf area index
clumping	clumping index
parameter	parameter array according to land cover types
meteo	meteorological data
CosZs	cosine of solar zenith angle
var_o	temporary variables array of last time step
var_n	temporary variables array of this time step
soilp	soil coefficients according to land cover types and soil textures
mid_res	results struct

Returns

void

5.12 meteo_pack.c File Reference

This function will calculate all the meteorological variables based on input.

```
#include "beps.h"
```

Functions

- void meteo_pack (double temp, double rh, double *meteo_pack_output)

  Function to calculate meteorological variables based on input.

5.12.1 Detailed Description

This function will calculate all the meteorological variables based on input.

Author

Edited by XZ Luo

Date

May 19, 2015

5.12.2 Function Documentation

5.12.2.1 meteo_pack()

```
void meteo_pack (
            double temp,
            double rh,
            double * meteo_pack_output )
```

Function to calculate meteorological variables based on input.
default input is temperature (C) and relative humidity (0-100) output is an array, named as meteo_pack_output []
[input] meteo_pack_output [1]= air_density kg/m3
meteo_pack_output [2]= specific heat of air J/kg/C
meteo_pack_output [3]= VPD kPa
meteo_pack_output [4]= slope of vapor pressure to temperature kPa/C
meteo_pack_output [5]= psychrometer constant kPa/C
meteo_pack_output [6]= saturate water vapor potential kPa
meteo_pack_output [7]= actual water vapor potential kPa
meteo_pack_output [8]= specific humidity g/g

Parameters

temp	temperature
rh	relative humidity
meteo_pack_output	meteorological variables array

Returns

void

5.13 netRadiation.c File Reference

This module will calculate net radiation at both canopy level and leaf level.

```
#include "beps.h"
```

Functions

- void netRadiation (double shortRad_global, double CosZs, double temp_o, double temp_u, double temp_g, double lai_o, double lai_u, double lai_os, double lai_us, double lai_o_sunlit, double lai_o_shaded, double lai_u_sunlit, double lai_u_shaded, double clumping, double temp_air, double rh, double albedo_snow_↩v, double albedo_snow_n, double percentArea_snow_o, double percentArea_snow_u, double percent_↩snow_g, double albedo_v_o, double albedo_n_o, double albedo_v_u, double albedo_n_u, double albedo_↩v_g, double albedo_n_g, double *netRad_o, double *netRad_u, double *netRad_g, double *netRadLeaf↩_o_sunlit, double *netRadLeaf_o_shaded, double *netRadLeaf_u_sunlit, double *netRadLeaf_u_shaded, double *netShortRadLeaf_o_sunlit, double *netShortRadLeaf_o_shaded, double *netShortRadLeaf_u_↩sunlit, double *netShortRadLeaf_u_shaded)

    Function to calculate net radiation at canopy level and leaf level.

5.13.1 Detailed Description

This module will calculate net radiation at both canopy level and leaf level.

Author

Edited by XZ Luo

Date

May 23, 2015

5.13.2 Function Documentation

5.13.2.1 netRadiation()

```
void netRadiation (
            double shortRad_global,
            double CosZs,
            double temp_o,
            double temp_u,
            double temp_g,
            double lai_o,
            double lai_u,
            double lai_os,
            double lai_us,
            double lai_o_sunlit,
            double lai_o_shaded,
            double lai_u_sunlit,
            double lai_u_shaded,
            double clumping,
            double temp_air,
            double rh,
            double albedo_snow_v,
            double albedo_snow_n,
            double percentArea_snow_o,
            double percentArea_snow_u,
            double percent_snow_g,
            double albedo_v_o,
            double albedo_n_o,
            double albedo_v_u,
            double albedo_n_u,
            double albedo_v_g,
            double albedo_n_g,
            double * netRad_o,
            double * netRad_u,
            double * netRad_g,
            double * netRadLeaf_o_sunlit,
            double * netRadLeaf_o_shaded,
            double * netRadLeaf_u_sunlit,
            double * netRadLeaf_u_shaded,
            double * netShortRadLeaf_o_sunlit,
            double * netShortRadLeaf_o_shaded,
            double * netShortRadLeaf_u_sunlit,
            double * netShortRadLeaf_u_shaded )
```

Function to calculate net radiation at canopy level and leaf level.
[input] global solar radiation, cosine value for solar zenith angle, albedo of leaves albedo of snow, percentage of snow cover, leaf area index overstorey and understorey, temperature of overstorey, understorey and ground (contain snow?) temperature of air (C), relative humidity (0-100)
[output] net radiation for canopy, overstorey, understorey and ground; net radiation on sunlit, shaded leaves of overstorey and understorey.

Parameters

shortRad_global	global short radiation
CosZs	cosine value of solar zenith angle
temp_o	temperature of overstorey
temp_u	temperature of understory

Parameters

temp_g	temperature of ground
lai_o	leaf area index of overstory, without stem
lai_u	leaf area index of understory, without stem
lai_os	leaf area index of overstory, with stem
lai_us	leaf area index of understory, with stem
lai_o_sunlit	sunlit leaves LAI with consideration of stem, overstory
lai_o_shaded	shaded leaves LAI with consideration of stem, overstory
lai_u_sunlit	sunlit leaves LAI with consideration of stem, understory
lai_u_shaded	shaded leaves LAI with consideration of stem, understory
clumping	clumping index
temp_air	air temperature
rh	relative humidity
albedo_snow_v	albedo of snow in this step, visible
albedo_snow_n	albedo of snow in this step, near infrared
percentArea_snow_o	percentage of snow on overstorey (by area)
percentArea_snow_u	percentage of snow on understorey (by area)
percent_snow_g	percentage of snow on ground (by mass)
albedo_v_o	albedo of overstory, visible, not considering snow, decided by land cover
albedo_n_o	albedo of overstory, near infrared
albedo_v_u	albedo of understory, visible
albedo_n_u	albedo of understory, near infrared
albedo_v_g	albedo of ground, visible
albedo_n_g	albedo of ground, near infrared
netRad_o	net radiation on overstorey
netRad_u	net radiation on understorey
netRad_g	net radiation on ground
netRadLeaf_o_sunlit	net radiation at the leaf level, overstory sunlit, for ET calculation
netRadLeaf_o_shaded	net radiation at the leaf level, overstory shaded
netRadLeaf_u_sunlit	net radiation at the leaf level, understory sunlit
netRadLeaf_u_shaded	net radiation at the leaf level, understory shaded
netShortRadLeaf_o_sunlit	net shortwave radiation at leaf level, overstory sunlit, for GPP calculation
netShortRadLeaf_o_shaded	net shortwave radiation at leaf level, overstory shaded
netShortRadLeaf_u_sunlit	net shortwave radiation at leaf level, understory sunlit
netShortRadLeaf_u_shaded	net shortwave radiation at leaf level, understory shaded

Returns

void

5.14 photosyn_gs.c File Reference

This program solves a cubic equation to calculate leaf photosynthesis.

```
#include "beps.h"
#include "DB.h"
```

Functions

- void photosynthesis (double temp_leaf_p, double rad_leaf, double e_air, double g_lb_w, double vc_opt, double f_soilwater, double b_h2o, double m_h2o, double cii, double temp_leaf_c, double LH_leaf, double *Gs_w, double *aphoto, double *ci)

  Function to calculate leaf photosynthesis by solving a cubic equation.
- double SFC_VPD (double temp_leaf_K, double leleafpt)

  This function computes the relative humidity at the leaf surface for application in the Ball Berry Equation. Latent heat flux, LE, are passed through the function, mol m-2 s-1, and it solves for the humidity at leaf surface.
- double TEMP_FUNC (double rate, double eact, double tprime, double tref, double t_lk)

  Arhennius temperature function.
- double LAMBDA (double tak)

  Function to calculate latent heat of vaporization in J kg-1.
- double ES (double t)

  Function to calculate saturation vapor pressure function in mb.
- double TBOLTZ (double rate, double eakin, double topt, double tl)

  Maxwell-Boltzmann temperature distribution for photosynthesis.

5.14.1 Detailed Description

This program solves a cubic equation to calculate leaf photosynthesis.

Author

W. Ju

Date

Jan 14, 1999

5.14.2 Function Documentation

5.14.2.1 ES()

```
double ES (
            double t )
```

Function to calculate saturation vapor pressure function in mb.

Parameters

t	temperature in Kelvin

Returns

double

5.14.2.2 LAMBDA()

```
double LAMBDA (
            double tak )
```

Function to calculate latent heat of vaporization in J kg-1.

Parameters

tak	

Returns

double

5.14.2.3 photosynthesis()

```
void photosynthesis (
            double temp_leaf_p,
            double rad_leaf,
            double e_air,
            double g_lb_w,
            double vc_opt,
            double f_soilwater,
            double b_h2o,
            double m_h2o,
            double cii,
            double temp_leaf_c,
            double LH_leaf,
            double * Gs_w,
            double * aphoto,
            double * ci )
```

Function to calculate leaf photosynthesis by solving a cubic equation.
[output] stomatal conductance to water vapor (m s-1); net photosynthesis rate (umol CO2 m-2 s-1); intercellular co2 concentration (ppm)

Parameters

temp_leaf←_p	temporary variables, to be removed later
rad_leaf	net shortwave radiation (W/m2)
e_air	water vapor pressure above canopy (kPa)
g_lb_w	leaf laminar boundary layer conductance to H2O (m/s)
vc_opt	the maximum velocities of carboxylation of Rubisco at 25 deg C (umol m-2 s-1)
f_soilwater	an empirical scalar of soil water stress on stomatal conductance, dimensionless
b_h2o	the intercept term in BWB model (mol H2O m-2 s-1)
m_h2o	the slope in BWB model
cii	initial intercellular co2 concentration (ppm)
temp_leaf←_c	leaf temperature (deg C)
LH_leaf	leaf latent heat flux (W m-2)
Gs_w	stomatal conductance to water vapor (m s-1)
aphoto	net photosynthesis rate (umol CO2 m-2 s-1)
ci	intercellular co2 concentration (ppm)

Returns

void

5.14.2.4 SFC_VPD()

```
double SFC_VPD (
            double temp_leaf_K,
            double leleafpt )
```

This function computes the relative humidity at the leaf surface for application in the Ball Berry Equation. Latent heat flux, LE, are passed through the function, mol m-2 s-1, and it solves for the humidity at leaf surface.

Parameters

temp_leaf_K	leaf temporary temperature in Kalvin
leleafpt	leaf latent heat

Returns

[rhum_leaf] humidity at leaf surface

double

5.14.2.5 TBOLTZ()

```
double TBOLTZ (
            double rate,
            double eakin,
            double topt,
            double tl )
```

Maxwell-Boltzmann temperature distribution for photosynthesis.

Parameters

rate	
eakin	
topt	
tl	

Returns

double

5.14.2.6 TEMP_FUNC()

```
double TEMP_FUNC (
            double rate,
            double eact,
            double tprime,
            double tref,
            double t_lk )
```

Arhennius temperature function.

Parameters

rate	the pre-exponential factor
eact	
tprime	
tref	reference temperature
t_lk	

Returns

double

5.15 plant_respir.c File Reference

Estimate plant respiration.

```
#include "beps.h"
```

Functions

- void plantresp (int LC, struct results *mid_res, double lai_yr, double lai, double temp_air, double temp_soil, double CosZs)

  Function to calculate plant respiration.

5.15.1 Detailed Description

Estimate plant respiration.

Authors

Written by: J. Liu. and W. Ju

Modified by G. Mo

Date

Last update: May 2015

5.15.2 Function Documentation

5.15.2.1 plantresp()

```
void plantresp (
            int LC,
            struct results * mid_res,
            double lai_yr,
            double lai,
            double temp_air,
            double temp_soil,
            double CosZs )
```

Function to calculate plant respiration.

Parameters

LC	land cover type
mid_res	results struct
lai_yr	annual mean leaf area index
lai	daily leaf area index
temp_air	air temperature
temp_soil	soil temperature
CosZs	cosine of solar zenith angle

Returns

void

5.16 rainfall.c File Reference

This module will calculate the water remained on canopy surface after evaporation in this step (used for next step)
`#include "beps.h"`

Functions

- void rainfall_stage1 (double temp_air, double precipitation, double mass_water_o_last, double mass_water_u_last, double lai_o, double lai_u, double clumping, double ∗mass_water_o, double ∗mass_water_u, double ∗percent_water_o, double ∗percent_water_u, double ∗precipitation_g)

  Function of rainfall stage1.
- void rainfall_stage2 (double evapo_water_o, double evapo_water_u, double ∗mass_water_o, double ∗mass_water_u)

  Function of rainfall stage2.

5.16.1 Detailed Description

This module will calculate the water remained on canopy surface after evaporation in this step (used for next step)
[rainfall_stage1] happens before evaporation of intercepted water from canopy (supply)
[rainfall_stage2] happens after evaporation of intercepted water from canopy (demand)

Note

rainfall on ground is considered in stage1, and then considered in surface water module (or soil moisture module)

Author

XZ Luo

Date

May 25, 2015

5.16.2 Function Documentation

5.16.2.1 rainfall_stage1()

```
void rainfall_stage1 (
            double temp_air,
            double precipitation,
            double mass_water_o_last,
            double mass_water_u_last,
            double lai_o,
            double lai_u,
            double clumping,
            double * mass_water_o,
            double * mass_water_u,
            double * percent_water_o,
            double * percent_water_u,
            double * precipitation_g )
```

Function of rainfall stage1.

[rainfall_stage1] happens before evaporation of intercepted water from canopy (supply)
[input] air temperature, precipitation (m/s), remain of water on leaves from last step (kg/m2) per leaf area leaf area index of overstorey and understorey, excluding stem. length of this step (s), if time step is 10min, then it is set as 600, air temperature and humidity
[output] percentage of canopy covered by rainfall, overstorey and understorey (provided to evaporation_canopy), mass of water available for evaporation on canopy in this step precipitation on ground
[optical output] intercepted mass of rainfall in this step

Parameters

temp_air	air temperature (Celsius)
precipitation	precipitation rate (m/s)
mass_water_o_last	remains of water from last step, overstory
mass_water_u_last	remains of water from last step, understory
lai_o	leaf area index, overstory
lai_u	leaf area index, understory
clumping	clumping index
mass_water_o	mass of water on leaves (kg/m2) per ground area, overstory
mass_water_u	mass of water on leaves (kg/m2) per ground area, understory
percent_water_o	the fraction of canopy covered by liquid water and snow, overstory
percent_water_u	the fraction of canopy covered by liquid water and snow, understory
precipitation_g	precipitation on ground

Returns

void

5.16.2.2 rainfall_stage2()

```
void rainfall_stage2 (
            double evapo_water_o,
            double evapo_water_u,
            double * mass_water_o,
            double * mass_water_u )
```

Function of rainfall stage2.
[rainfall_stage2] happens after evaporation of intercepted water from canopy (demand)
[input] mass of water on leaves after precipitation in this step, evaporation from leaves in this step
[output] mass of water on leaves after the evaporation on leaves in this step (this value is transferred to next step)

Parameters

evapo_water_o	evaporation of intercepted rain in this step, overstorey, kg/m2/s = mm/s
evapo_water_u	evaporation of intercepted rain in this step, understorey, kg/m2/s = mm/s
mass_water_o	supply of rain on leaves, overstory, already added precipitation in this step
mass_water_u	supply of rain on leaves, understory, already added precipitation in this step

Returns

void

5.17 readcoef.c File Reference

Set soil coefficients according to land cover types and soil types for soil respiration and NEP calculation.

```
#include "beps.h"
```

Functions

- void readcoef (int short lc, int stxt, double ∗coef)

  Function to set soil coefficients.

5.17.1 Detailed Description

Set soil coefficients according to land cover types and soil types for soil respiration and NEP calculation.

Author

G. Mo

Date

Dec., 2005

5.17.2 Function Documentation

5.17.2.1 readcoef()

```
void readcoef (
            int short lc,
            int stxt,
            double * coef )
```

Function to set soil coefficients.

Parameters

lc	land cover type 1-ENF 2-DNF 6-DBF 9-EBF 13-Shrub 40-C4 Plants default:Others
stxt	soil texture
coef	soil coefficients array

Returns

void

5.18 readparam.c File Reference

Set parameters according to land cover types.

```
#include "beps.h"
```

Functions

- void readparam (short lc, double ∗parameter1)

Function to set parameters.

5.18.1 Detailed Description

Set parameters according to land cover types.

Author

Gang Mo

Date

Apr., 2011

5.18.2 Function Documentation

5.18.2.1 readparam()

```
void readparam (
            short lc,
            double * parameter1 )
```

Function to set parameters.

Parameters

lc	land cover type 1-ENF 2-DNF 6-DBF 9-EBF 13-Shrub 40-C4 Plants default-Others
parameter1	parameter array

Returns

void

5.19 s_coszs.c File Reference

calculate cos_solar zenith angle Z

```
#include "beps.h"
```

Functions

- void s_coszs (short jday, short j, float lat, float lon, double *CosZs)

  Function to calculate cosine solar zenith angle.

5.19.1 Detailed Description

calculate cos_solar zenith angle Z

Author

W. Ju

Date

July, 2004

5.19.2 Function Documentation

5.19.2.1 s_coszs()

```
void s_coszs (
            short jday,
            short j,
            float lat,
            float lon,
            double * CosZs )
```

Function to calculate cosine solar zenith angle.

Parameters

jday	date of year
j	local time/UTC time code needs to be edited according to time format
lat	latitude of site
lon	longitude of site
CosZs	cosine solar zenith angle

Returns

void

5.20 sensible_heat.c File Reference

This module will calculate sensible heat from overstorey, understorey and ground.

`#include "beps.h"`

Functions

- void sensible_heat (double tempL_o_sunlit, double tempL_o_shaded, double tempL_u_sunlit, double tempL_u_shaded, double temp_g, double temp_air, double rh_air, double Gheat_o_sunlit, double Gheat_o_shaded, double Gheat_u_sunlit, double Gheat_u_shaded, double Gheat_g, double lai_o_sunlit, double lai_o_shaded, double lai_u_sunlit, double lai_u_shaded, double *SH_o, double *SH_u, double *SH_g)

  Function to calculate sensible heat.

5.20.1 Detailed Description

This module will calculate sensible heat from overstorey, understorey and ground.

Author

Edited by XZ Luo

Date

May 23, 2015

5.20.2 Function Documentation

5.20.2.1 sensible_heat()

```
void sensible_heat (
            double tempL_o_sunlit,
            double tempL_o_shaded,
            double tempL_u_sunlit,
```

```
double tempL_u_shaded,
double temp_g,
double temp_air,
double rh_air,
double Gheat_o_sunlit,
double Gheat_o_shaded,
double Gheat_u_sunlit,
double Gheat_u_shaded,
double Gheat_g,
double lai_o_sunlit,
double lai_o_shaded,
double lai_u_sunlit,
double lai_u_shaded,
double * SH_o,
double * SH_u,
double * SH_g )
```

Function to calculate sensible heat.
[input] temperature of sunlit and shaded leaves from other storey (leaf temperature module); temperature of air; relative humidity; temperature of ground (soil heat flux module); aerodynamic heat conductance of sunlit shaded leaves from overstorey and understorey; aerodynamic heat conductance of ground; leaf area index, sunlit and shaded, overstorey and understorey (from leaf area index module);
[output] sensible heat from overstorey, understorey and ground

Parameters

tempL_o_sunlit	temperature of leaves, overstory, sunlit
tempL_o_shaded	temperature of leaves, overstory, shaded
tempL_u_sunlit	temperature of leaves, understory, sunlit
tempL_u_shaded	temperature of leaves, understory, shded
temp_g	temperature of ground
temp_air	air temperature
rh_air	relative humidity of air
Gheat_o_sunlit	aerodynamic resistance of heat, overstory, sunlit
Gheat_o_shaded	aerodynamic resistance of heat, overstory, shaded
Gheat_u_sunlit	aerodynamic resistance of heat, understory, sunlit
Gheat_u_shaded	aerodynamic resistance of heat, understory, shaded
Gheat_g	aerodynamic resistance of heat, ground
lai_o_sunlit	leaf area index, overstory, sunlit
lai_o_shaded	leaf area index, overstory, shaded
lai_u_sunlit	leaf area index, understory, sunlit
lai_u_shaded	leaf area index, understory, shaded
SH_o	sensible heat, overstory
SH_u	sensible heat, understory
SH_g	sensible heat, ground

Returns

void

5.21 snowpack.c File Reference

This module will calculate the percentage of canopy and ground covered by snow and output albedo of snow (used in energy balance) and density of snow in this step.

```
#include "beps.h"
```

Functions

- void snowpack_stage1 (double temp_air, double precipitation, double mass_snow_o_last, double mass_↩ snow_u_last, double mass_snow_g_last, double ∗mass_snow_o, double ∗mass_snow_u, double ∗mass↩ _snow_g, double lai_o, double lai_u, double clumping, double ∗area_snow_o, double ∗area_snow_u, double ∗percent_snow_o, double ∗percent_snow_u, double ∗percent_snow_g, double ∗density_snow, double ∗depth_snow, double ∗albedo_v_snow, double ∗albedo_n_snow)

  Function of snowpack stage1.
- void snowpack_stage2 (double evapo_snow_o, double evapo_snow_u, double ∗mass_snow_o, double ∗mass_snow_u)

  Function of snowpack stage2. This module will calculate the snow remained on canopy surface after evaporation in this step.
- void snowpack_stage3 (double temp_air, double temp_snow, double temp_snow_last, double density_snow, double ∗depth_snow, double ∗depth_water, double ∗mass_snow_g)

  Function of snowpack stage3. This module simulates the process of snow melting and water frozen in this step.

5.21.1 Detailed Description

This module will calculate the percentage of canopy and ground covered by snow and output albedo of snow (used in energy balance) and density of snow in this step.
[snowpack_stage1] happens before any consumption of snow in this step, after the snow fall (supply)
[snowpack_stage2] happens after sublimation from ground and canopy (demand)
[snowpack stage3] happens after frozen and melt of snow pack (demand)

Author

XZ Luo

Date

May 25, 2015

5.21.2 Function Documentation

5.21.2.1 snowpack_stage1()

```
void snowpack_stage1 (
            double temp_air,
            double precipitation,
            double mass_snow_o_last,
            double mass_snow_u_last,
            double mass_snow_g_last,
            double * mass_snow_o,
            double * mass_snow_u,
            double * mass_snow_g,
            double lai_o,
            double lai_u,
            double clumping,
            double * area_snow_o,
            double * area_snow_u,
            double * percent_snow_o,
            double * percent_snow_u,
            double * percent_snow_g,
            double * density_snow,
```

```
            double * depth_snow,
            double * albedo_v_snow,
            double * albedo_n_snow )
```

Function of snowpack stage1.
[snowpack_stage1] happens before any consumption of snow in this step, after the snow fall (supply)
[Input] air temperature, precipitation,depth of snow from last step, density of snow from last step, mass of snow on canopy and ground (per ground area) from last step, length of step, leaf area index of overstorey and understorey excluding stem, albedo of snow from last step.
[Output] mass of snow on canopy and ground accumulation of snowfall, albedo of snow in this step, density of snow in this step.

Parameters

temp_air	air temperature
precipitation	precipitation (m/s)
mass_snow_o_last	weight of snow at overstorey from last step
mass_snow_u_last	weight of snow at understorey from last step
mass_snow_g_last	weight of snow on ground from last step
mass_snow_o	mass of intercepted snow at overstory, input from last step, kg/m2
mass_snow_u	mass of intercepted snow at understory, input from last step, kg/m2
mass_snow_g	mass of intercepted snow on ground, input from last step, kg/m2
lai_o	overstory lai
lai_u	understory lai
clumping	clumping index
area_snow_o	area of snow at overstorey
area_snow_u	area of snow at understorey
percent_snow_o	percentage of snow cover at overstory, DECIDED by weight
percent_snow_u	percentage of snow cover at understory, DECIDED by weight
percent_snow_g	percentage of snow cover on ground, DECIDED by weight
density_snow	density of snowpack on ground, input from last step, then changed in this module
depth_snow	depth of snowpack, input from last step, changed here, then changed in stage2
albedo_v_snow	visible albedo of snow, input from this step, changed in this module
albedo_n_snow	near infrared albedo of snow, input from this step, changed in this module

Returns

void

5.21.2.2 snowpack_stage2()

```
void snowpack_stage2 (
            double evapo_snow_o,
            double evapo_snow_u,
            double * mass_snow_o,
            double * mass_snow_u )
```

Function of snowpack stage2. This module will calculate the snow remained on canopy surface after evaporation in this step.
[snowpack_stage2] happens after sublimation from ground and canopy (demand)
[input] mass of snow on leaves after precipitation in this step, sublimation from leaves in this step
[output] mass of snow on leaves after the sublimation on leaves in this step

Parameters

evapo_snow↔_o	evaporation of intercepted rain in this step, overstorey, kg/m2/s = mm/s
evapo_snow↔_u	evaporation of intercepted rain in this step, understorey, kg/m2/s = mm/s
mass_snow↔_o	supply of rain on leaves, overstorey, already added precipitation in this step
mass_snow↔_u	supply of rain on leaves, understorey, already added precipitation in this step

Returns

void

5.21.2.3 snowpack_stage3()

```
void snowpack_stage3 (
            double temp_air,
            double temp_snow,
            double temp_snow_last,
            double density_snow,
            double * depth_snow,
            double * depth_water,
            double * mass_snow_g )
```

Function of snowpack stage3. This module simulates the process of snow melting and water frozen in this step.
[snowpack stage3] happens after frozen and melt of snow pack (demand)
[input] depth of snow on ground after stage 1, air temperature, ground surface temperature
[output] the amount of the melted snow, frozen snow

Parameters

temp_air	temperature of air in this step
temp_snow	temperature of snow in this step
temp_snow_last	temperature of snow in last step
density_snow	density of snow output from stage1
depth_snow	depth of snow on ground after stage1
mass_snow_g	mass of snow on ground after stage1
depth_water	depth of water after all precipitation and evaporation

Returns

void

5.22 soil.h File Reference

Header file for soil struct.

Classes

- struct Soil

    Define soil struct.

Macros

- #define **FW_VERSION** 1
- #define **max**(a, b) ((a)>(b))?(a):(b)
- #define **min**(a, b) ((a)<(b))?(a):(b)
- #define **MAX_LAYERS** 10
- #define **DEPTH_F** 6

Functions

- void SoilRootFraction (struct Soil soil[ ])

  Declare functions.
- void Init_Soil_Parameters (int landcover, int stxt, double r_root_decay, struct Soil p[ ])

  Function to initialize soil parameters.
- void Init_Soil_Status (struct Soil p[ ], double Tsoil, double Tair, double Ms, double snowdepth)

  Function to initialize the soil status: soil temperature and moisture for each layer, ponded water, snow depth, et al.
- void soil_water_factor_v2 (struct Soil p[ ])

  Function to compute soil water stress factor.
- void Soil_Water_Uptake (struct Soil p[ ], double Trans_o, double Trans_u, double Evap_soil)

  Function to calcualte soil water uptake from a layer.
- void **UpdateSoilLambda** (struct Soil soil[ ])
- void **init_soil_parameter** (unsigned char T_USDA, unsigned char S_USDA, unsigned char Ref_Depth, double T_Density, double S_Density, double T_OC, double S_OC, struct Soil soil[ ])
- void Update_Cs (struct Soil p[ ])

  Function to update volume heat capacity.
- void Update_ice_ratio (struct Soil p[ ])

  Function to update the frozen status of each soil.
- void UpdateSoilThermalConductivity (struct Soil p[ ])

  Function to update soil thermal conductivity.
- void UpdateHeatFlux (struct Soil p[ ], double Xg_snow, double lambda_snow, double Tsn0, double Tair_↩annual_mean, double peroid_in_seconds)

  Function to update soil heat flux.
- void UpdateSoilMoisture (struct Soil p[ ], double peroid_in_seconds)

  Function to update soil moisture.

5.22.1 Detailed Description

Header file for soil struct.

Author

Liming He

Date

Dec. 05, 2012

Last revision on May 15, 2015

5.22.2 Function Documentation

5.22.2.1 Init_Soil_Parameters()

```
void Init_Soil_Parameters (
            int landcover,
            int stxt,
            double r_root_decay,
            struct Soil p[] )
```

Function to initialize soil parameters.
[1] Set the depth for each layer
[2] Set the parameters for each layer

Parameters

landcover	land cover type
stxt	soil texture
r_root_decay	decay rate of root distribution
p	Soil struct variable

Returns

void

5.22.2.2 Init_Soil_Status()

```
void Init_Soil_Status (
            struct Soil p[],
            double Tsoil,
            double Tair,
            double Ms,
            double snowdepth )
```

Function to initialize the soil status: soil temperature and moisture for each layer, ponded water, snow depth, et al.

Parameters

p	Soil struct variable
Tsoil	soil temperature
Tair	air temperature
Ms	soil water content
snowdepth	snow depth

Returns

void

5.22.2.3 soil_water_factor_v2()

```
void soil_water_factor_v2 (
            struct Soil p[] )
```

Function to compute soil water stress factor.
[output] dt, fw-soil water stress

Parameters

p	soil conditions struct

Returns

void

5.22.2.4 Soil_Water_Uptake()

```
void Soil_Water_Uptake (
            struct Soil p[ ],
            double Trans_o,
            double Trans_u,
            double Evap_soil )
```

Function to calcualte soil water uptake from a layer.

Parameters

p	soil variables struct
Trans_o	transpiration from overstory canopies
Trans_u	transpiration from understory canopies
Evap_soil	evaporation from soil

Returns

void

5.22.2.5 SoilRootFraction()

```
void SoilRootFraction (
            struct Soil soil[ ] )
```

Declare functions.
Declare functions.

Parameters

soil	Soil struct variable

Returns

void

5.22.2.6 Update_Cs()

```
void Update_Cs (
            struct Soil p[ ] )
```

Function to update volume heat capacity.

Parameters

p	soil variables struct

Returns

void

5.22.2.7 Update_ice_ratio()

```
void Update_ice_ratio (
            struct Soil p[] )
```

Function to update the frozen status of each soil.

Parameters

p	soil variables struct

Returns

void

5.22.2.8 UpdateHeatFlux()

```
void UpdateHeatFlux (
            struct Soil p[],
            double Xg_snow,
            double lambda_snow,
            double Tsn0,
            double Tair_annual_mean,
            double period_in_seconds )
```

Function to update soil heat flux.

Parameters

p	soil variables struct
Xg_snow	the fraction of the ground surface covered by snow
lambda_snow	the effective thermal conductivity of snow –in m^2/s
Tsn0	surface temperature
Tair_annual_mean	annual mean air temperature
period_in_seconds	360 sec. per time, 10 times per hour

Returns

void

5.22.2.9 UpdateSoilMoisture()

```
void UpdateSoilMoisture (
            struct Soil p[],
            double kstep )
```

Function to update soil moisture.

Parameters

p	soil variables struct
kstep	the total seconds in this step (period), defined in beps.h

Note

kkk (outside of the function): step within an hour or half hour measurement

Returns

void

5.22.2.10 UpdateSoilThermalConductivity()

```
void UpdateSoilThermalConductivity (
                struct Soil p[] )
```

Function to update soil thermal conductivity.
[input] thermal_cond, fei, ice_ratio, thetam, kw, ki
[output] lambda for each layer

Parameters

p	soil variables struct

Returns

void

5.23 soil.h

Go to the documentation of this file.

```

#ifndef SOIL_H
#define SOIL_H

#define FW_VERSION 1 // 0 for soil water uptake using R, and 1 for soil water uptake using R*fpsisr

#define max(a,b)     ((a)>(b))?(a):(b)
#define min(a,b)     ((a)<(b))?(a):(b)

// Note: change the value of MAX_LAYERS to a small one for global application
// e.g. max layers = 6.
#define MAX_LAYERS 10 // LHE. Jan 28, 2013.
#define DEPTH_F 6

struct Soil{
    /*********************************************/
    // Properties belong to the whole soil profile
    int flag;                // reserved for EnKF usage.
    int n_layer;             // the number of layers used in the model. Make sure n_layer <= MAX_LAYERS
    int step_period;

    // Conditions on the top boundary
    double Zp;               // depth of ponded water on the ground surface
    double Zsp;              // snow depth
    double r_rain_g;         // the rainfall rate, un--on understory g--on ground surface  m/s
    double soil_r;           // soil surface resistance for water, discuss with Remi - an interface here
    double r_drainage;

    // Some variable used for soil
    // double t1;
    // double t2;
    double r_root_decay;     // decay_rate_of_root_distribution
    double psi_min;          // for fw
    double alpha;            // for fw
    double f_soilwater;

    /*********************************************/
    // Properties belong to each soil horizon
    double d_soil[MAX_LAYERS];
    double f_root[MAX_LAYERS];        // root weight
    double dt[MAX_LAYERS];            // the weight calculated from soil_water_factor **re-calculate in the model

    // From read-param function
```

```
    double thermal_cond[MAX_LAYERS]; // thermal conductivity. Unit:
    double theta_vfc[MAX_LAYERS];    // field capacity (not used in this model. LHE. Feb. 01, 2013)
    double theta_vwp[MAX_LAYERS];    // wilt point*/
    double fei[MAX_LAYERS];          // porosity */
    double Ksat[MAX_LAYERS];         // saturated hydraulic conductivity
    double psi_sat[MAX_LAYERS];      // water potential at sat
    double b[MAX_LAYERS];            // Cambell parameter b
    double density_soil[MAX_LAYERS]; // soil bulk density of layer. LHE. Feb. 12, 2013.
    double f_org[MAX_LAYERS];        // volume fraction of organic matter in layer (%).

    // Variables need to save
    double ice_ratio[MAX_LAYERS];    // the ratio of ice of soil layer
    double thetam[MAX_LAYERS], thetam_prev[MAX_LAYERS]; // soil water content in this layer

    // soil temperature in this layer, don't change because it is used in soil_water_factor_v2, and UpdateSoil_Moisture.
    double temp_soil_p[MAX_LAYERS];
    // soil temperature in this layer. don't change because it is used in soil_water_factor_v2, and UpdateSoil_Moisture.
    double temp_soil_c[MAX_LAYERS];


    // Derived variables below:
    double f_ice[MAX_LAYERS];        // derived var.
    double psim[MAX_LAYERS];         // soil water suction in this layer. Note: this variable can be derived from other parameters. LHE.
    double thetab[MAX_LAYERS];       // soil water content at the bottom of each layer
    double psib[MAX_LAYERS];         // soil water suction at the bottom this layer
    double r_waterflow[MAX_LAYERS];  // the liquid water flow rates at the soil layer interfaces  'eg. 0,1,2..., represents the surface, the bottom of layer1, the bottom of layer2,...

    double km[MAX_LAYERS], Kb[MAX_LAYERS];   //the hydraulic conductivity of certain soil layer
    double KK[MAX_LAYERS];           // The average  conductivity of two soil layers.*/

    double Cs[MAX_LAYERS];
    double lambda[MAX_LAYERS];       // thermal conductivity of each soil layer /* ={0} by LHE */ // not used in gpp-only version. derived var.
    double Ett[MAX_LAYERS];          // ET in each layer. derived var

    // define a lambda_top for ice?
    double G[MAX_LAYERS];            // energy fluxes
};

void SoilRootFraction(struct Soil soil[]);
void Init_Soil_Parameters(int landcover, int stxt, double r_root_decay, struct Soil p[]);
void Init_Soil_Status(struct Soil p[], double Tsoil, double Tair, double Ms, double snowdepth);
void soil_water_factor_v2(struct Soil p[]);
void Soil_Water_Uptake(struct Soil p[], double Trans_o, double Trans_u, double Evap_soil);
void UpdateSoilLambda(struct Soil soil[]);
void init_soil_parameter(unsigned char T_USDA, unsigned char S_USDA, unsigned char Ref_Depth, double
    T_Density, double S_Density, double T_OC, double S_OC, struct Soil soil[]);
void Update_Cs(struct Soil p[]);
void Update_ice_ratio(struct Soil p[]);
void UpdateSoilThermalConductivity(struct Soil p[]);
void UpdateHeatFlux(struct Soil p[], double Xg_snow, double lambda_snow, double Tsn0, double
    Tair_annual_mean, double peroid_in_seconds);
void UpdateSoilMoisture(struct Soil p[], double peroid_in_seconds);

#endif
```

5.24 soil_thermal_regime.c File Reference

Soil thermal regime: update the soil temperature fore each soil layer.

```
#include "soil.h"
#include <math.h>
```

Functions

- void UpdateHeatFlux (struct Soil p[ ], double Xg_snow, double lambda_snow, double Tsn0, double Tair_annual_mean, double period_in_seconds)

    Function to update soil heat flux.

- void Update_Cs (struct Soil p[ ])

    Function to update volume heat capacity.

- void Update_ice_ratio (struct Soil p[ ])

    Function to update the frozen status of each soil.

- void UpdateSoilThermalConductivity (struct Soil p[ ])

  Function to update soil thermal conductivity.

5.24.1 Detailed Description

Soil thermal regime: update the soil temperature fore each soil layer.

Author

Liming He

Date

Last update: Sept. 15, 2015

5.24.2 Function Documentation

5.24.2.1 Update_Cs()

```
void Update_Cs (
        struct Soil p[ ] )
```

Function to update volume heat capacity.

Parameters

p	soil variables struct

Returns

void

5.24.2.2 Update_ice_ratio()

```
void Update_ice_ratio (
        struct Soil p[ ] )
```

Function to update the frozen status of each soil.

Parameters

p	soil variables struct

Returns

void

5.24.2.3 UpdateHeatFlux()

```
void UpdateHeatFlux (
        struct Soil p[ ],
        double Xg_snow,
        double lambda_snow,
        double Tsn0,
        double Tair_annual_mean,
        double period_in_seconds )
```

Function to update soil heat flux.

Parameters

p	soil variables struct
Xg_snow	the fraction of the ground surface covered by snow
lambda_snow	the effective thermal conductivity of snow –in m^2/s
Tsn0	surface temperature
Tair_annual_mean	annual mean air temperature
period_in_seconds	360 sec. per time, 10 times per hour

Returns

void

5.24.2.4 UpdateSoilThermalConductivity()

```
void UpdateSoilThermalConductivity (
            struct Soil p[ ] )
```

Function to update soil thermal conductivity.
[input] thermal_cond, fei, ice_ratio, thetam, kw, ki
[output] lambda for each layer

Parameters

p	soil variables struct

Returns

void

5.25 soil_water_stress.c File Reference

Compute soil water stress factor.

```
#include <stdio.h>
#include <math.h>
#include "debug.h"
#include "soil.h"
```

Functions

- void soil_water_factor_v2 (struct Soil p[ ])

  Function to compute soil water stress factor.

5.25.1 Detailed Description

Compute soil water stress factor.

Note

Please refer to file soil_water_factor.cpp for the original version. LHE.

Version

2.0

Authors

Rewritten by: Liming He. Jan 29, 2013.

Modified by: Mustapha El Maayar. March 2008

Written by: Weimin Ju

Last revision by: Liming He.

Date

May 22, 2015.

5.25.2 Function Documentation

5.25.2.1 soil_water_factor_v2()

```
void soil_water_factor_v2 (
            struct Soil p[] )
```

Function to compute soil water stress factor.
[output] dt, fw-soil water stress

Parameters

p	soil conditions struct

Returns

void

5.26 soilresp.c File Reference

This module is to calculate soil respiration.

```
#include "beps.h"
#include "soil.h"
```

Functions

- void soilresp (double ∗Ccd, double ∗Cssd, double ∗Csmd, double ∗Cfsd, double ∗Cfmd, double ∗Csm, double ∗Cm, double ∗Cs, double ∗Cp, float npp_yr, double ∗coef, int soiltype, struct Soil ∗soilp, struct results ∗mid←_res)

    Function to calculate soil respiration.

5.26.1 Detailed Description

This module is to calculate soil respiration.

5.26.2 Function Documentation

5.26.2.1 soilresp()

```
void soilresp (
            double * Ccd,
            double * Cssd,
            double * Csmd,
            double * Cfsd,
            double * Cfmd,
            double * Csm,
            double * Cm,
            double * Cs,
            double * Cp,
            float npp_yr,
            double * coef,
            int soiltype,
            struct Soil * soilp,
            struct results * mid_res )
```

Function to calculate soil respiration.

Parameters

Ccd	carbon pool variable
Cssd	...
Csmd	...
Cfsd	...
Cfmd	...
Csm	...
Cm	...
Cs	...
Cp	...
npp_yr	a fraction of NPP transferred to biomass carbon pools
coef	soil coefficients array
soiltype	soil type
soilp	soil variables struct
mid_res	results struct

Returns

void

NEP

5.27 surface_temp.c File Reference

This module will simulate surface temperature in each step, as well as heat flux form surface to soil layers.

```
#include "beps.h"
```

Functions

- void surface_temperature (double temp_air, double rh_air, double depth_snow, double depth_water, double capacity_heat_soil1, double capacity_heat_soil0, double Gheat_g, double depth_soil1, double density_snow, double tempL_u, double netRad_g, double evapo_soil, double evapo_water_g, double evapo_snow_g, double lambda_soil1, double percent_snow_g, double heat_flux_soil1, double temp_ground_last, double temp_soil1_last, double temp_any0_last, double temp_snow_last, double temp_soil0_last, double temp↩_snow1_last, double temp_snow2_last, double ∗temp_ground, double ∗temp_any0, double ∗temp_snow, double ∗temp_soil0, double ∗temp_snow1, double ∗temp_snow2, double ∗heat_flux)

Function to simulate surface temperature, and heat flux from surface to soil layers.

5.27.1 Detailed Description

This module will simulate surface temperature in each step, as well as heat flux form surface to soil layers.
As it is an interface between ground, air and soil, the core idea is to separate the interface as different layers by depth of snow, then calculate the temperature gradient and at last calculate the heat flux from ground surface to soil.
Original beps would use Xg_snow[kkk] at some places after snow melt & frozen, now we uniformly use the value before snow melt & frozen.

Author

Edited by XZ Luo

Date

June 1, 2015

5.27.2 Function Documentation

5.27.2.1 surface_temperature()

```
void surface_temperature (
            double temp_air,
            double rh_air,
            double depth_snow,
            double depth_water,
            double capacity_heat_soil1,
            double capacity_heat_soil0,
            double Gheat_g,
            double depth_soil1,
            double density_snow,
            double tempL_u,
            double netRad_g,
            double evapo_soil,
            double evapo_water_g,
            double evapo_snow_g,
            double lambda_soil1,
            double percent_snow_g,
            double heat_flux_soil1,
            double temp_ground_last,
            double temp_soil1_last,
            double temp_any0_last,
            double temp_snow_last,
            double temp_soil0_last,
            double temp_snow1_last,
            double temp_snow2_last,
            double * temp_ground,
            double * temp_any0,
            double * temp_snow,
            double * temp_soil0,
            double * temp_snow1,
            double * temp_snow2,
            double * heat_flux )
```

Function to simulate surface temperature, and heat flux from surface to soil layers.

Parameters

temp_air	air temperature (Celsius degree)
rh_air	relative humidity (0-100)
depth_snow	depth of snow (m)
depth_water	depth of water on ground (m)
capacity_heat_soil1	heat capacity of layer1 soil (J/m2/K)
capacity_heat_soil0	heat capacity of layer2 soil (J/m2/K)
Gheat_g	aerodynamic conductance of heat on ground (m/s)
depth_soil1	depth of soil in layer1 (m)
density_snow	density of snow (kg/m3)
tempL_u	leaf temperature, understory (Celsius degree)
netRad_g	net radiation on ground (W/m2)
evapo_soil	evaporation from soil surface (mm/s)
evapo_water_g	evaporation from pond water on ground (mm/s)
evapo_snow_g	evaporation from snow pack on ground (mm/s)
lambda_soil1	thermal conductivity of layer1 soil (W/m/K)
percent_snow_g	percentage of snow coverage on ground (0-1)
heat_flux_soil1	heat flux from layer1 soil to the next soil layer (W/m2)
temp_ground_last	temperature of ground, from last step
temp_soil1_last	temperature of layer1 soil, from last step
temp_any0_last	temperature of any layer right above the soil, from last step
temp_snow_last	temperature of snow, from last step
temp_soil0_last	temperature of soil0, from last step
temp_snow1_last	temperature of snow layer 2, from last step
temp_snow2_last	temperature of snow layer 3, from last step
temp_ground	ground temperature at this step
temp_any0	temperature of any layer right above the soil could be a mixture of snow temperature and soil surface temperature
temp_snow	snow temperature at this step
temp_soil0	temperature of soil surface right above the soil, the part not covered by snow
temp_snow1	temperature of snow layer 2, used in depth_snow>0.05 m
temp_snow2	temperature of snow layer 3, used in depth_snow>0.05 m
heat_flux	heat flux from ground to soil

Returns

void

5.28 transpiration.c File Reference

This module calculates transpiration, for overstorey and understorey, sunlit and shaded.

```
#include "beps.h"
```

Functions

- void transpiration (double tempL_o_sunlit, double tempL_o_shaded, double tempL_u_sunlit, double tempL_u_shaded, double temp_air, double rh_air, double Gtrans_o_sunlit, double Gtrans_o_shaded, double Gtrans_u_sunlit, double Gtrans_u_shaded, double lai_o_sunlit, double lai_o_shaded, double lai_u_sunlit, double lai_u_shaded, double *trans_o, double *trans_u)

Function to calculate transpiration.

5.28.1 Detailed Description

This module calculates transpiration, for overstorey and understorey, sunlit and shaded.

Author

Edited by XZ Luo

Date

May 20, 2015

5.28.2 Function Documentation

5.28.2.1 transpiration()

```
void transpiration (
            double tempL_o_sunlit,
            double tempL_o_shaded,
            double tempL_u_sunlit,
            double tempL_u_shaded,
            double temp_air,
            double rh_air,
            double Gtrans_o_sunlit,
            double Gtrans_o_shaded,
            double Gtrans_u_sunlit,
            double Gtrans_u_shaded,
            double lai_o_sunlit,
            double lai_o_shaded,
            double lai_u_sunlit,
            double lai_u_shaded,
            double * trans_o,
            double * trans_u )
```

Function to calculate transpiration.
A transformation of Penman-Monteith equation is used here. It could be regarded as a mass transfer process. Water vapor inside cells are required by VPD from air and VPD on leaf surface.
[input] temperature of sunlit and shaded leaves from other storey (leaf temperature module); temperature of air; relative humidity; conductance of water for sunlit shaded leaves from overstorey and understorey; leaf area index, sunlit and shaded, overstorey and understorey (from leaf area index module);
[output] transpiration from overstorey and understorey

Parameters

tempL_o_sunlit	temperature of leaf, overstory, sunlit
tempL_o_shaded	temperature of leaf, overstory, shaded
tempL_u_sunlit	temperature of leaf, understory, sunlit
tempL_u_shaded	temperature of leaf, understory, shaded
temp_air	air temperature
rh_air	relative humidity of air
Gtrans_o_sunlit	total conductance of water tandem of stomatal conductance and aerodynamic conductance, overstory, sunlit
Gtrans_o_shaded	..., overstory, shaded
Gtrans_u_sunlit	..., understory, sunlit
Gtrans_u_shaded	..., understory, shaded

Parameters

lai_o_sunlit	leaf area index, overstory, sunlit
lai_o_shaded	leaf area index, overstory, shaded
lai_u_sunlit	leaf area index, understory, sunlit
lai_u_shaded	leaf area index, understory, shaded
trans_o	transpiration from overstory
trans_u	transpiration from understory

Returns

void

5.29 updatesoilmoisture.c File Reference

This module will calculate soil moisture after a period, given the current condition.

```
#include "soil.h"
#include <math.h>
```

Functions

- void UpdateSoilMoisture (struct Soil p[ ], double kstep)

  Function to update soil moisture.
- void Soil_Water_Uptake (struct Soil p[ ], double Trans_o, double Trans_u, double Evap_soil)

  Function to calcualte soil water uptake from a layer.

5.29.1 Detailed Description

This module will calculate soil moisture after a period, given the current condition.
Based on Richards equation, sources: ET and rain

Author

Last revision: L. He

Date

May 20, 2015

5.29.2 Function Documentation

5.29.2.1 Soil_Water_Uptake()

```
void Soil_Water_Uptake (
            struct Soil p[ ],
            double Trans_o,
            double Trans_u,
            double Evap_soil )
```

Function to calcualte soil water uptake from a layer.

Parameters

p	soil variables struct
Trans_o	transpiration from overstory canopies
Trans_u	transpiration from understory canopies
Evap_soil	evaporation from soil

Returns

void

5.29.2.2 UpdateSoilMoisture()

```
void UpdateSoilMoisture (
            struct Soil p[],
            double kstep )
```

Function to update soil moisture.

Parameters

p	soil variables struct
kstep	the total seconds in this step (period), defined in beps.h

Note

kkk (outside of the function): step within an hour or half hour measurement

Returns

void