Estimation and Validation of Carbon/Water Cycles

in a Mongolian Grassland Ecosystem under Non-grazing Condition

Using Sim-CYCLE

January, 2006

Lee PILZAE

Estimation and Validation of Carbon/Water Cycles in a Mongolian Grassland Ecosystem under Non-grazing Condition Using Sim-CYCLE

A Dissertation Submitted to
the Graduate School of Life and Environmental Sciences,
the University of Tsukuba
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
(Doctoral Program in Integrative Environmental Sciences)

Lee PILZAE

Contents

Abstract iii

Abbreviations v

List of Figures ix

List of Tables x

1. Introduction 1

2. Method 3

2.1 Model description 3

2.1.1 Brief overview of Sim-CYCLE 3

2.1.1.1 Carbon cycle 3

2.1.1.2 Water cycle 6

2.1.2 Modification improvement of process in Sim-CYCLE 8

2.1.2.1 Translocation from root to shoot 8

2.1.2.2 Change in number of soil-water layer 8

2.2 Study site and data description 9

2.3 Initial value and parameterization 11

2.4 Simulation 12

3. Results 13

3.1 Annual carbon/water cycles and validation 13

3.2 Monthly carbon/water cycles and validation 13

3.3 Future carbon/water dynamics 15

4. Discussion 17

5. Conclusions 22

Acknowledgements 23

References 24

Appendix: Program list A-1

Abstract

Two processes of an ecosystem model, Sim-CYCLE (Ito and Oikawa, 2002) have been improved in order for reflecting a Mongolian steppe well. One was the change of structure in soil-water availability (the change of soil-water layers from 2 to 3) and the other was translocation process from root to shoot (considering the ratio of aboveground biomass to belowground biomass). The parameters of the improved Sim-CYCLE were calibrated with ecological data measured under non-grazing condition at Kherlenbayann-Ulaan (KBU) in Mongolia and previous literature. Annual average results of this calibrated Sim-CYCLE showed a high consistence with measured data, such as aboveground biomass (AB), belowground biomass (BB), heterotrophic respiration (HR), lower soil moisture content (MS_{LW}), and evapotranspiration (ET).

Future carbon/water dynamics in this site was simulated under four environmental change scenarios based on past meteorological data during 11 years, which showed a decreasing trend of precipitation (about 20 mm decrease) and an increasing trend of air temperature (about 0.5℃ increase) in growing-season (May-October): 1) S1P: To decrease 1% precipitation annually every year until 100 years (from 245 to 90 mm annually), 2) S2T: To increase 0.025℃ annually in every growing-season until 100 years (from 0.3℃ to 2.8℃ annually), 3) S3C: To increase 1% atmospheric CO_2 concentration annually until 100 years (from 378 to

698 ppm), and 4) S4A: To sum the conditions of S1P, S2T, and S3C. The predicted results at the 100th year from these scenarios are as follows: From the S1P scenario, NPP, AB, BB, and transpiration (TR) were estimated to decrease up to 81, 79, 76, and 94% compared with those of the first year, due to water stress to photosynthesis, respectively. From the S2T scenario, those were estimated to decrease 32, 31, 28, and 47%, respectively, due to temperature stress to photosynthesis. From the S3C scenario, those were estimated to inversely increase 55, 53, 50, and 22%, respectively, because of CO_2 fertilization effect. Lastly, from the S4A scenario, those were estimated to decrease 68, 66, 63, and 92%, respectively, because of both water and temperature stress to photosynthesis, that is, the decrease of soil water content (47%) had an effect on stomatal closure, and then led to the decrease of transpiration (92%). The simulations imply that KBU will be vulnerable under the change of precipitation and temperature as well as suggests that the research site may have a high possibility on ecosystem deterioration after around 100 years under similar climatic change undergone until now even under non-grazing condition.

Abbreviations

Term	Definition (Units)
AD	air density (kg m^{-3})
AB	aboveground biomass (Mg C ha^{-1})
ANPP	aboveground net primary productivity (Mg C ha^{-1} $month^{-1}$)
AR	autotrophic respiration (Mg C ha^{-1} $month^{-1}$)
ARG	autotrophic growth respiration (Mg C ha^{-1} $month^{-1}$)
ARM	autotrophic maintenance respiration (Mg C ha^{-1} $month^{-1}$)
ASH	specific heat of air (=0.2813 J kg^{-1} $℃^{-1}$)
BB	belowground biomass (Mg C ha^{-1})
BNPP	belowground net primary productivity (Mg C ha^{-1} $month^{-1}$)
CV	convexity of water availability-evapotranspiration rate curve (dimensionless)
DI	dryness index related to runoff (mm)
DL	day-length (hours)
EP	effective photosynthate for biomass growth [=*GPP* – *ARM*] (C ha^{-1} $month^{-1}$)
ET	evapotranspiration rate (mm $month^{-1}$)
EV	evaporation rate (mm $month^{-1}$)
GA	aerodynamic conductance (mmol H_2O m^{-2} s^{-1})

GG	ground conductance for evaporation from non-saturated soil (mm H_2O m^{-2} s^{-1})
GPP	gross primary productivity (Mg C ha^{-1} $month^{-1}$)
GR	grazing (Mg C ha^{-1} $month^{-1}$)
GS	leaf stomatal conductance (mmol CO_2 m^{-2} s^{-1})
HR	heterotrophic respiration (Mg C ha^{-1} $month^{-1}$)
IC	interception (mm $month^{-1}$)
KA	light attenuation coefficient (dimensionless)
LAI	leaf area index (m^2 m^{-2})
LF	litter fall (Mg C ha^{-1} $month^{-1}$)
LH	specific latent heat of water vaporization (=2.5 MJ kg^{-1} H_2O)
MS_D	deep soil moisture content (mm)
MS_{LW}	lower soil moisture content (mm)
MS_{UP}	upper soil moisture content (mm)
NEP	net ecosystem productivity (Mg C ha^{-1} $month^{-1}$)
NPP	net primary productivity (Mg C ha^{-1} $month^{-1}$)
PC	single-leaf photosynthetic rate (μmol CO_2 m^{-2} s^{-1})
PC_{SAT}	light-saturated photosynthetic rate (μmol CO_2 m^{-2} s^{-1})
PEV	potential evaporation rate (mm $month^{-1}$)

$PPFD_{TOP}$	photosynthetic photon flux density at the canopy-top (μmol photons m^{-2} s^{-1})
PSC	psychrometer constant (=0.667 Pa K^{-1})
PTR	potential transpiration rate (mm $month^{-1}$)
QE	quantum yield, photochemical light-use efficiency (mol CO_2 mol^{-1}photon)
RNC	net radiation at canopy (W m^{-2})
RNS	net radiation at soil surface (W m^{-2})
RO	runoff (mm mon^{-1})
$SARG_{F,C,R}$	specific growth respiration coefficient (Mg C Mg C^{-1})
$SARM_{F,C,R}$	specific maintenance respiration rate (Mg C Mg C^{-1} day^{-1})
$SLF_{F,C,R}$	specific litter fall rate (Mg C Mg C^{-1} day^{-1})
$SLVP_{SAT}$	slope of the saturated vapor pressure-temperatur curve (hPa ℃$^{-1}$)
$T_{OPT,MIN,MAX}$	optimum, minimum, and maximum temperature for photosynthesis (℃)
T_R	translocation rate (dimensionless)
T_{RS}	translocation from root to shoot (Mg C ha^{-1})
TP	proportion of root biomass translocated to shoot (dimensionless)
TR	transpiration rate (mm $month^{-1}$)
VPD	vapor pressure deficit of air (hPa)
WHC	water holding capacity of soil (mm)

$WP_{F,C,R}$	foliage, stem, and root biomass (Mg C ha^{-1})
$WS_{L,H}$	litter and mineral soil carbon storage (Mg C ha^{-1})

List of Figures

Figure	Title	Page
1	Annual mean air temperature and precipitation in KBU during 1993-2003	32
2	Meteorological data of KBU at 2003	33
3	Linear regression analysis of measured and simulated LAI	34
4	Monthly change of measured and simulated aboveground biomass (a) and belowground biomass (b)	35
5	Monthly change of measured and simulated evapotranspiration (a) and MS_{LW} (b)	36
6	Results of future carbon/water cycles in S1P (a, b), S2T (c, d), S3C (e, f), and S4A (g, h) during 100 years under non-grazing condition	37

List of Tables

Table	Title	Page
1	Air temperature from 1993-2003 at KBU	38
2	Surface temperature from 1993-2003 at KBU	39
3	Precipitation from 1993-2003 at KBU	40
4	List of parameters used in the model	41
5	Comparison between the measurements and simulated results at KBU	42
6	Terns of the energy, water, and carbon budgets for 2003	43
7	Results of future carbon/water cycles after 100 years	44

1. Introduction

Mongolia is located at northeastern Asia and has around 1.56 x 10^6 km^2, where a forest-grassland-desert ecotone is formed along a steep gradient in the geographical and climatic condition. The ecotone is a sensitive transitional area between two adjacent ecological types and generally sensitive for climate and vegetation change (Di Castri, 1988). The largest biome is steppe, covering 83.4% of the total land area of the country, which corresponds to roughly 2.6% of the global grassland vegetation (World Resources Institute, 2003). The steppe is ecologically fragile and sensitive to seasonal and decadal changes in climate. Droughts are usually the most significant limiting factor for grass growth in such semiarid grassland ecosystems (Seligman & Van Leulen, 1989; Wilhite, 1993; Seastedt et al., 1998).

The climate of northeast Asia (China, Russia, and Mongolia) has undergone significant changes over the last 30 years (1979-1997, +1.5℃; Chase et al., 2000). Recent findings (Chase et al., 2000; Mongolian Action Program, 2000) show that this region has one of the strongest warming signals on the earth. Analysis of the meteorological records shows that over last 60 years spring rainfall in the Mongolian steppes has declined by 17%, mostly in May (Natsagdorj, 2000; Baatarbileg et al., 2001). Model projections suggest that there will be a warmer and drier climate in the semi-arid and arid regions of Asia in the future if the levels of atmospheric greenhouse gases continue to increase (McCarthy et al., 2001). It is expected that the climate

change may considerably affect Mongolian steppe in the future while it in turn may have the potential to feed back to regional and possibly global climate as a result of its large area (Smith et al., 1996)

Despite large area of Mongolia and sensitivity of plant under climatic change, the interaction of atmosphere-vegetation in this steppe was poorly understood. Also, considering local and global role of this area under climate change, it is urgent to understand carbon/water cycles. To these requests, RAISE (Rangelands Atmosphere-hydrosphere-biosphere Interaction Study Experiment in Northeastern Asia) project has been undertaken with the overall aims of examining eco-hydrological processes as well as the effects on regional and global carbon/water cycles at the research sites (forest and typical steppe) of Kherlen river basin in Mongolia from 2003.

Mongolia steppes generally include grazing, but this study was conducted under non-grazing condition as the first step of understanding the carbon/water cycles of a Mongolian steppe. The aims of this research are: 1) to improve some processes of Sim-CYCLE reflecting a Mongolian steppe well under non-grazing condition; 2) to calibrate parameters of the Sim-CYCLE, and; 3) to estimate potential future carbon/water dynamics under environmental change conditions after conducting validation of the Sim-CYCLE.

2. Method

2.1 Model description

2.1.1 Brief overview of Sim-CYCLE

2.1.1.1 Carbon cycle

A process-based model, Sim-CYCLE (Ito and Oikawa, 2002) was used to simulate the carbon dynamics of a Mongolian steppe. The model has five compartments: foliage (subscript F), stem (subscript C), root (subscript R), litter (subscript L), and humus (subscript H). Total amount of carbon in a given ecosystem (WE) is composed of plant biomass (WP) and soil organic carbon (WS). WP is distributed in foliage, stem, and root; WS is distributed in litter and mineral soil.

$$WE=WP+WS$$

$$WP=WP_F+WP_C+WP_R$$

$$WS=WS_L+WS_H \qquad (1)$$

In this study, we did not divide WP_F and WP_C. Thus, aboveground biomass (AB) was defined by WP_F+WP_C and the notation of belowground biomass (BB) was used for substitution of WP_R for convenience.

Net primary production (NPP) and net ecosystem production (NEP) were defined through three major processes of CO_2 exchange between atmosphere and biosphere which are gross primary production (GPP), autotrophic plant respiration (AR), grazing (GR), and heterotrophic

soil respiration (HR) as:

$$NPP = GPP - AR - GR \quad (2)$$

$$NEP = NPP - HR \quad (3)$$

Also, NPP can be divided by aboveground NPP (ANPP) and belowground NPP (BNPP).

$$NPP = ANPP + BNPP \quad (4)$$

The instantaneous GPP_{INS} rate was calculated using the dry-matter production theory established by Monsi and Saeki (1953):

$$\begin{aligned} GPP_{INS} &= \int_0^{LAI} PC\, dLAI \\ &= \frac{PC_{SAT}}{KA}\left[\ln\left\{QE + KA \cdot PPFD_{TOP}\right\}\right] \\ &\quad - \ln\left\{QE + KA \cdot PPFD_{TOP} \cdot \exp\left(-KA \cdot LAI\right)\right\} \end{aligned} \quad (5)$$

where PC is single-leaf photosynthetic rate, LAI leaf area index, PC_{SAT} the single-leaf photosynthetic rate under light-saturation, QE light-use efficiency, KA light attenuation coefficient, and $PPFD_{TOP}$ the photosynthetic photon flux density at the canopy top.

According to Kuroiwa (1966), the diurnal change in $PPFD_{TOP}$ is approximated well by a sine-square curve with the peak at midday, as follows:

$$PPFD_{TOP} = PPFD_{MD} \sin^2\left(\frac{360 \cdot t}{DL}\right), \quad (6)$$

where t is time since sunrise, DL is day length, and $PPFD_{MD}$ is the irradiance of $PPFD_{TOP}$ at midday. After substituting $PPFD_{TOP}$, we integrate GPP_{INS} for DL to obtain the daily GPP_{DAY} rate.

$$GPP_{DAY} = \varepsilon \int_0^{DL} \int_0^{LAI} PC\, dLAI dt$$

$$= \frac{2\varepsilon PC_{SAT}\, DL}{KA} \left[\ln\left\{ 1 + \sqrt{1 + \frac{KA \cdot QE \cdot PPFD_{MD}}{PC_{SAT}}} \right\} \right]$$

$$- \ln\left\{ 1 + \sqrt{1 + \frac{KA \cdot QE \cdot PPFD_{MD} \cdot \exp(-KA \cdot LAI)}{PC_{SAT}}} \right\} \tag{7}$$

where ε is the unit conversion factor (=4.32 × 10^{-4}).

AR is composed of two components that have distinct functional meanings for maintenance and for growth (Amthor, 1989). AR is calculated as a sum of six respiration rates:

$$AR = ARM + ARG$$

$$AR = \sum_{X=organ}^{F,C,R} ARM_X + \sum_{X=organ}^{F,C,R} ARG_X , \tag{8}$$

where ARM denotes the maintenance respiration of the whole plant, foliage, stem, and root, and similarly ARG denotes growth respiration. ARM is a function of the amount of existing carbon (WP) and temperature (TG):

$$ARM_X = SARM_X \exp\left[\frac{\ln QT}{10} (TG - 15) \right] WP_X , \tag{9}$$

where SARM is the specific respiration rate and TG is the temperature (control temperature, 15 °C). QT represents the sensitivity to temperature change. ARG is an explicit function of plant growth rate. Thus, ARG is calculated only when biomass has a net gain:

$$ARG_X = SARG_X \Delta WP_X , \tag{10}$$

where SARG is the specific growth respiration rate. $SARG_{F,C}$ and $SARG_R$ are approximately 0.25 (McCree, 1970; Amthor, 1984; Ruimy et al., 1996) and 0.2 (Bachelet, 1989), respectively

HR is composed of two constituents from each compartment and is affected by temperature and soil moisture condition.

$$HR = HR_L + HR_H \tag{11}$$

Litter fall (LF) process is one of the most difficult processes for mechanistic models to simulate. In this study, constant mortality or turnover rate irrespective of environmental conditions is assumed during growing season. Most of plants' shoots are shed if temperature is less then 5 °C while the roots have a constant mortality all the year round:

$$LF = \sum_{X=organ}^{F,C,R} \left(SLF_X \cdot WP_X \right), \tag{12}$$

where SLF is the specific litter-fall rate. SLF was changed in this study based on measured litter fall data (Lui, 2004; Urano et al., 2004).

2.1.1.2 Water cycle

Sim-CYCLE also has water/radiation submodel. Precipitation is partitioned into soil-water availability (MS), evaporation from the soil surface (EV) and canopy surface (IC), transpiration from the canopy (TR), and subsurface runoff (RO). The MS is divided by snow accumulation (SNA), upper MS (MS_{UP}: 0-30cm) and lower MS (MS_{LW}: 30-200cm or rooting depth). The TR and EV depend on the energy inputs to the canopy (RNC) and soil (RNG), and occur in MS_{LW} and MS_{UP}, respectively. Potential evaporation (PEV) and transpiration (PTR) are estimated by

the Penman–Monteith method, assuming an abundant water supply.

$$PEV = \frac{SLVP_{SAT} \cdot RNG + AD \cdot ASH \cdot VPD \cdot GG}{LH\left[SLVP_{SAT} + PSC(GG/GA)\right]} \tag{13}$$

$$PTR = \frac{SLVP_{SAT} \cdot RNC + AD \cdot ASH \cdot VPD \cdot GS}{LH\left[SLVP_{SAT} + PSC(GS/GA)\right]}, \tag{14}$$

where AD is air density, ASH is the specific heat of air, LH is the latent heat of vaporization (= 2.5 MJ kg^{-1} H_2O), and PSC is the psychrometer constant (= 0.667 Pa kg^{-1}). $SLVP_{SAT}$ is the slope of saturated vapor pressure which is a function of temperature, and VPD is the vapor pressure deficit. GG and GS denote the ground conductance for water vapor and single-leaf stomatal conductance, respectively. EV and TR are approximated as a solution of quadratic function due to the limitation of water availability:

$$EV = \frac{\left(MS_{UP} + PEV\right) - \sqrt{\left(MS_{UP} + PEV\right)^2 - 4CV \cdot MS_{UP} \cdot PEV}}{2CV} \tag{15}$$

$$TR = \frac{\left(MS_{LW} + PTR\right) - \sqrt{\left(MS_{LW} + PTR\right)^2 - 4CV \cdot MS_{LW} \cdot PTR}}{2CV}, \tag{16}$$

where CV is the convexity of the MS_{UP}-EV and MS_{LW}-TR curves (= 0.85 in this study)

The runoff (RO) from the MS_{LW} is parameterized by using a simple bucket-model equation (Manabe, 1969):

$$RO = \sqrt[3]{MS_{LW}{}^3 + DI^3} - DI \tag{17}$$

$$DI = WHC - MS_{LW}, \tag{18}$$

where, DI denotes the dryness index and WHC is the soil water holding capacity.

2.1.2 Modification improvement of process in Sim-CYCLE

2.1.2.1 Translocation from root to shoot

In Sim-CYCLE, translocation of carbohydrates from roots to shoots, T_{RS} occurs only at emergency period. Translocation process from root to shoot proposed by Hanson et al. (1988), which necessitates three conditions for this process to occur: 1) Soil temperature must be greater than 5.5℃; 2) MS_{LW} must be greater than 7%, and; 3) BB > 20 * AB. If these three conditions are satisfied:

$$T_{RS} = TP * BB \quad (19)$$

where TP is the proportion of root biomass translocated monthly to shoots (= 0.01 in this study).

2.1.2.2 Change in number of soil-water layer

In Sim-CYCLE, there are 2 soil-water layers, which are MS_{UP} (0-30cm) and MS_{LW} (30-200cm or rooting depth). However, this soil-water structure could not reflect real one of the Mongolian steppe well because most of root biomass was distributed in the depth of 0-50cm (Liu, 2004). Thus, we increased soil-water layer from 2 to 3, that is, MS_{UP} (0-5cm), MS_{LW} (5-50cm), and MS_D (50-200cm or rooting depth). Evaporation and transpiration occur in MS_{UP} and MS_{LW}, respectively. The hydrometric conductance which goes from MS_{UP} to MS_{LW} and MS_{LW} to MS_D

was set to static value, 0.4 and 0.34 monthly, respectively. Runoff occurs only in MS_D.

2.2 Study site and data description

The research site is located in Kherlenbayan-Ulaan (KBU), which is located in Hentiy province of Mongolia (47°12N, 108°44E; 1235m above sea level). The climatic characteristics were cold and dry in winter and warm in summer as well as large difference between day and night temperatures. During 1993-2003, the mean annual air temperature and soil surface temperature were 1.1℃ and 3.2℃, respectively. The mean annual precipitation was 201 mm and was concentrated from June to September (about 88%). The pattern of precipitation was quite irregular by year and showed strong seasonal variability. The growing season was normally from late April to mid October when the monthly mean temperature was above 5℃. Meteorological data of KBU are shown in Tables 1, 2, and 3 from a local meteorological station of the Institute of Meteorology and Hydrology of Mongolia and provided for the Rangelands Atmosphere-hydrosphere-biosphere Interaction Study Experiment in Northeastern Asia (RAISE) from 1993 to 2003. Meteorological data in 2003 was shown in Fig. 2. The characteristic of air temperature was warm in summer and cold in winter with the peak in July (22.8℃). Most precipitation (99%) was concentrated on plant growing season (late April to middle October), which showed a typical monsoon. Especially, the precipitation during June to

September was 85% of annual precipitation. The peak of precipitation occurred at September (65mm), leading to the peak of soil water content (Fig. 9). Net radiation of the steppe was measured from 25 March 2003 to 24 March 2004. Global radiation and net radiation were 5574 and 1563 MJ m^{-2} annually and peak at June (263 W m^{-2}) and July (119 W m^{-2}), respectively.

The vegetation was typical short-grass steppe, which had the average height of 15 cm according to a 2-year field data. Dominant species were *Stipa krylovii* and *Artemisia frigida* for C_3 and *Cleistogenes squarrosa* for C_4. C_3 plants occupied about 90% (Mariko *et al*., 2003). The soil was a typical Chestnut soil.

A non-grazing experiment plot (200 × 170 m) was fenced with a height of 1.5 m in August 2002 for evaluating the grazing effect on the steppe. Next year (2003), field experiments were carried out inside and outside of the non-grazing experiment plot every month from June to September. Mariko et al. (2003) and Urano et al. (2004) measured LAI, above-ground biomass (AB), and heterotrophic respiration (HR) with ecological methods, such as harvesting method, camber method. Liu et al. (2004) measured below-ground biomass (BB) and below-ground NPP (BNPP) by the soil boring stick method and the ingrowth soil core method, respectively. Li et al. (2005) estimated and measured gross ecosystem productivity (GEP), total ecosystem respiration (R_{eco}), and net ecosystem exchange (NEE) by the eddy covariance method.

2.3 Initial value and parameterization

It is essential to know initial values of a given ecosystem such as biomass, mineral soil, and soil water content in order for the simulation of carbon/water cycle. In this study, we used initial values measured in KBU site in 2003: 0 Mg C ha^{-1} for AB, 5.5 Mg C ha^{-1} for BB, 58.40 Mg C ha^{-1} for WS_H, 0 Mg C ha^{-1} for WS_L, 2.0% for MS_{UP}, 3.3% for MS_{LW}, and 4.9% for MS_D. There was no WS_L data in January 2003; we assumed that this value was 0. Initial value of BB was averaged with measured data of June, July, and September because of no data measured at January.

It is difficult for calibrating parameters of mechanical model because of many parameters as well as the insufficiency of measured data for conducting parametrization. Sim-CYCLE had 34 vegetations defined by Olson et al. (1983) and the parameters in each vegetation were calibrated on 4 sites. Despite a lot of parameters of Sim-CYCLE, the measured data of this research site (non-grazing) showed that the parameters used in Sim-CYCLE needed to be improved in order for explaining carbon/water cycles in this site more exactly. This was because the parameterized sites conducted by Sim-CYCLE and this research site were different in some points: Leaf Area Index (LAI) of the validation sites in Sim-CYCLE was all over 1 m^2/ m^2, but that of this research site was less then 1 m^2/ m^2; The ratio of BB/AB in Sim-CYCLE was all less then 5, but that in this research site was over 10. Therefore, we calibrated parameters of Sim-CYCLE

adaptive to the measured data for the sake of reproducing the carbon/water cycles of this site better. Table 4 shows calibrated parameters. The parameters of respiration, litter fall, and optimum temperature for photosynthesis were mainly changed and the others were set to the same values used in Sim-CYCLE.

2.4 Simulation

After obtaining appropriate parameters in this site, we simulated the carbon/water cycles in 2003 with initial values. We also simulated future carbon/water dynamics in this site under four environmental change conditions based on meteorological data (Fig. 1), which had the characteristics of a decreasing trend of precipitation (about 20 mm decrease during 11 years) and an increasing trend of air temperature (about 0.5℃ increase during 11 years) in growing-season (May-October): 1) S1P: To decrease 1% precipitation annually every year until 100 years (from 245 to 90 mm annually), 2) S2T: To increase 0.025℃ annually in every growing-season until 100 years (from 0.3℃ to 2.8℃ annually), 3) S3C: To increase 1% CO_2 concentration annually until 100 years (from 378 to 698 ppm), and 4) S4A: To sum the conditions of S1P, S2T, and S3C. In S3C scenario, we followed A1B scenario of IPCC (2000).

3. Results

3.1 Annual carbon/water cycles and validation

In comparison with field data, we conclude that the simulated carbon/water dynamics was sufficiently close to the actual dynamics (Table 5). Relative errors of AB, BB, ANPP, BNPP, HR, and MS_{LW} were all less than 2%. The relative error of ET (6.8%) showed a relatively high difference owing to inclusion of grazing effect in measured data. Maximum LAI values were 0.58 and 0.55 for measurement and simulation, respectively; Difference of these two values was caused by monthly calculation of simulated LAI. Regression analysis showed that measured and simulated LAI agreed well with a coefficient of determination of 0.94 (Fig. 3). Table 6 shows simulation results for the carbon budget. ARM (24% of AR) was less than ARG (76% of AR). Belowground respiration depends on the amount of carbon allocation of roots because it is the sum of growth respiration and maintenance respiration. Despite the large ratio of BB/AB (= 12), belowground respiration was 48% of AR owing to the low maintenance respiration of roots.

3.2 Monthly carbon/water cycles and validation

Figure 4 (a) shows monthly change of measured and simulated AB. These values had similar patterns and high coefficient determination of 0.93. The period of plant growth was between April and October, which agreed with measurement. The fast growth at June and July resulted

from high photosynthesis rate due to favorable climate condition as well as translocation of carbohydrate from root to shoot (T_{RS} of June and July were 0.18, 0.17 Mg C ha^{-1} $month^{-1}$, respectively). High BB/AB ratio was resulted from this translocation. Peak AB occurred at August but growing rate was less than that of July because of no translocated carbohydrate from root to shoots as well as low photosynthesis rate compared to that of July, which was caused by mean surface temperature (19.7℃), soil water content (8%), and net radiation (271 MJ $month^{-1}$ m^{-2}) of August. In September, fast litter fall, amount of 0.27 Mg C ha^{-1} happened owing to decrease of air temperature (11.3℃) and produced carbohydrate was allocated to roots; Allocation to roots: 0.23 Mg C ha^{-1}, allocation to shoots: 0.10 Mg C ha^{-1}. Most AB shed their biomass after litter of 0.25 Mg C ha^{-1} was fallen at October.

Figure 4 (b) shows monthly change of measured and simulated BB, which was increasing trends during June to September. This was due to the allocation of photosynthate from shoots to roots and lower maintenance respiration of roots than its allocation. Liner regression analysis shows relatively low correspondence (R^2 = 0.58) because of small number of measurement data (n = 3) and large difference between measured (4.8 Mg C ha^{-1}) and simulated BB (5.3 Mg C ha^{-1}) of June. Both measured and simulated BB had minimum values at June; Simulated BB decreased from start of year. This phenomenon was due to maintenance respiration of roots and translocation carbohydrate from roots to young shoots. Simulated BB had a peak at September

owing to allocation to roots.

Measured and simulated monthly ET is shown in Fig. 5 (a), which illustrates a similar pattern and high correspondence with a coefficient of determination of 0.94 by linear regression analysis. ET at August showed the largest difference (10 mm $month^{-1}$) during one year due to high transpiration caused by the largest LAI. Peak of ET was at July owing to relatively large rainfall (55mm $month^{-1}$) as well as strongest net radiation (317 MJ $month^{-1}$ m^{-2}).

The increase of soil-water layers from 2 to 3 enabled us to properly simulate soil water content, especially, the variation of monthly MS_{LW}. Figure 5 (b) shows monthly change of measured and simulated MS_{LW}, showing a similar pattern and relatively high correspondence (R^2 = 0.81). The highest rainfall during 2003 appeared at September, so that MS_{LW} was the highest. This value was the lowest at April due to small precipitation and continuous penetration.

3.3 Future carbon/water dynamics

Table 7 summaries predicted carbon/water cycles after 100-year under non-grazing condition. The predicted results at the 100th year from these four scenarios were as follows: from S1P scenario, NPP, AB, BB, and transpiration (TR) were estimated to decrease up to 81, 79, 76, and 94% to the first year, due to water stress to photosynthesis, respectively. From the S2T scenario, those were estimated to decrease 32, 31, 28, and 47%, respectively, due to temperature stress to

photosynthesis. From the S3C scenario, those were estimated to inversely increase 55, 53, 50, and 22%, respectively because of CO_2 fertilization effect. Lastly, from the S4A scenario, those were estimated to decrease 68, 66, 63, and 92%, respectively because of both water and temperature stress to photosynthesis, that is, the decrease of soil water content (47%) had an effect on stomatal closure, and then led to the decrease of transpiration (92%).

Figure 6 shows results of future carbon/water cycles in S1P, S2T, S3C, and S4A scenarios during 100 years. This area was more sensitive for precipitation than temperature from the results of S1P and S2T scenarios (Fig. 6 a, b, c, and d). This agreed with the researches that precipitation is usually a limiting factor for grass growth in semiarid grassland ecosystems (Seligman & Van Leulen, 1989; Wilhite, 1993; Seastedt et al., 1998). In S1P, evaporation, transpiration, and MS_{LW} decreased because of decrease of precipitation. In S2T, increase of temperature caused increase of MS_{LW}: temperature stress caused stomatal closure, and then decreased the transpiration which occurred in MS_{LW} in the model. Thus, MS_{LW} decreased. It was notable that the productivity and plant biomass decreased faster as time passes at S4A scenario (Fig. 6 g): the productivity and biomass decreased 24% and 22% at 50th year, but 68% and 66% at 100th year, respectively.

4. Discussion

Mean annual ANPP and NPP in simulation were estimated as 0.45 and 1.31 Mg C ha^{-1} $year^{-1}$, respectively. Thus, ANPP/NPP ratio was 0.35 (c.f. Tables 5 and 6), which meant that growing rate of aboveground was just around one-second of that of belowground. The range of ANPP/NPP ratio is from 0.25 to 0.6 for semi-arid grasslands (Sims and Singh, 1978; Milchunas and Lauenroth, 1992). The present value (0.35) implicated that belowground was relatively important in this semi-arid grassland ecologically. Fifty-two percent of GPP and fifty-eight percent of EP (effective photosynthate for biomass growth) were allocated to roots. These values were smaller due to low maintenance respiration of roots (0.17 Mg C ha^{-1} $year^{-1}$) than the simulation results by Delting et al. (2000) who found that 65% of GPP and 80% of EP. The parameterization of low maintenance respiration of roots was based on measured data, which showed BB/AB ratio was over 10 under non-grazing condition (Liu, 2004). Root respiration represented 48% of AR. This is consistent with the simulation result by Nouvellon et al. (2000) and an intermediate value between the lowest 21% by Bachelet (1989) and the highest 71% by Detling et al. (1979).

Due to the low canopy fractional cover to ground (maximum LAI = 0.58 m^2/m^2), transpiration was generally less than evaporation. Transpiration, evaporation, and evapotranspiration (ET) were estimated to 42mm, 133mm, and 175mm, respectively.

Accordingly, evaporation was estimated to 76% of ET and transpiration to 24% of ET. This is consistent with the simulation results by Nouvellon et al. (2000) who simulated that evaporation and transpiration were 80% and 20% of ET in southeastern Arizona, respectively. On an annual basis, MS_{LW} was estimated as 6.3%. This low soil water content could explain the fact that precipitation is usually a limiting factor for plant growth in semiarid ecosystems (Seastedt et al., 1998). Generally, the rainfall patterns are of importance for perennial grass growth and survival. However, the improved Sim-CYCLE which had monthly time scale did not represent water stress to plant growth according to the event of precipitation which was highly variable day by day in this site, in spite of high correspondence between the measured and the modeled MS_{LW} (Fig. 5). For example, Li et al. (2005) showed that water stress was observed in late July to early August in the same site, but it was unable to realize this stress on this monthly based model.

Water use efficiency (WUE) indicates the amount of assimilated carbon per unit water transpired from the canopy. Thus, it is a useful indicator of the efficiency with which scarce water resources are used by plants in arid or semiarid environments. When WUE was considered as ANPP/ET in order for comparing WUE values given in the literature for natural ecosystems because of the difficulty of estimating BNPP and transpiration, this value in this model was 0.56 g DM kg^{-1} evapotranspirated H_2O. Values are ranging from 0.2 to 0.7 g DM kg^{-1} for shortgrass prairies of the United States (e.g. Webb et al., 1978; Lauenroth, 1979; Le

Houérou, 1984; Sala et al., 1988; Liang et al., 1989). When this value is defined as the ratio of NPP/ET, 1.62 g DM kg^{-1} evapotranspirated H_2O was estimated on an annual basis and was a relatively large value in the middle of the WUE values which is ranged between 0.87 and 2.07 g DM kg^{-1} evapotranspirated H_2O obtained by Sims and Singh (1978) on desert grassland. Le Houérou (1984) found that WUE tended to decrease when aridity increased and potential evaporation increased. This finding enabled us to understand the magnitude of aridity in this study site and in turn, the climate condition of this site was able to explain the low estimates of WUE, which was related to the low transpiration/ET ratio.

Future simulations showed that NPP and biomass were most sensitive to change in precipitation as time passes. However combinations of precipitation, temperature, and CO_2 had synergistic effects on NPP and biomass. Decreases in precipitation and increases in temperature had great negative effects on NPP and biomass due to water and temperature stress to plant, while increases in CO_2 had positive effects on them due to CO_2 fertilization effect. Decrease in aboveground biomass (AB) leaded to decease photosynthesis, and then brought to diminish translocation of photosynthate from shoot to root. Accordingly, belowground biomass (BB) was reduced. Because grazing generally decreases root growth due to the reduction of AB resulting from consumption by animals (Sundriyal 1992; Beaulieu et al. 1996; Biondini et al. 1998), it seems to be possible that larger BB will be decreased when considering grazing effect in this

site.

Previous models have projected grassland production under climate change (Coughenour and Parton, 1997; Gao and Zhang, 1997; Neilson et al., 1998; Christensen et al., 2004). Coughenour and Parton (1997) found elevated CO_2 at ambient and increased precipitation levels resulted in a 20-70% increase in C_3 grassland production. Gao and Zhang (1997) estimated a 15% increase in vegetation production with doubled CO_2 in the northeast region of China and Wand et al. (1999) concluded grass biomass on average increased by 44% with elevated CO_2. Christensen et al. (2004) used SAVANNA model and estimated a 78% decrease in productivity with the simulation scenario of doubled CO_2, decreased precipitation (80% of ambient precipitation), and increased temperature (2℃ increase in maximum temperature) at Inner Mongolia. Although these estimations had differences in magnitude and amount including the results of this study, the changes of pattern in productivity and biomass were similar under climate change, showing an increase in elevated CO_2, a decrease in increased temperature, and most of all a decrease sensitively in decreased precipitation.

Nomadic herding is the major human activity in Mongolia. About 20% of the human population is pastoralists and half of the population depends directly or indirectly on the pastoral economy for its livelihood (Fernandez-Gimenez & Allen-Diaz, 1999). Furthermore, human populations in Mongolia have increased continuously. This expansion of human

populations leads to increased pressures on rangeland for their resources (Polley et al., 2000). Excessive grazing is detrimental to plant communities (Cnoant & Paustian, 2002). After all, the change of land-use by grazing could have disastrous effects coupled with climate change (Archer, 1996). Models have been used to address the effect of grazing and climate on land use processes (Parton et al., 1987; Hanson et al., 1988; hunt et al., 1991; Coughenour and Chen, 1997; Ludwig et al., 2001; Christensen et al., 2004) but a lot of unsolved questions are remained (Nosberger et al., 2000). Despite simple assumption of climate change scenarios in this study, model simulations gave detailed insight as to which climate change scenarios had a large impact on carbon/water cycles. Mongolian steppe may have a high possibility on ecosystem deterioration after around 100 years under similar climatic change undergone until now even under non-grazing condition.

This study purposed to understand carbon/water cycles in a Mongolian steppe and was conducted under non-grazing condition as the first step for the aim. Thus, I have some plans afterward: (1) insertion of grazing effect in the improved Sim-CYCLE; (2) validation and parameterization with the field data measured in KBU at 2004 and 2005 for improvement of model confidence; (3) use of GCM model to overcome simple assumption of climate change scenarios.

5. Conclusion

Two processes of an ecosystem model, Sim-CYCLE have been improved in order for reflecting a Mongolian steppe well. One was the change of structure in soil-water availability (the change of soil-water layers from 2 to 3) and the other was translocation process from root to shoot. The improved Sim-CYCLE simulated the carbon and water cycles of a Mongolian steppe with a monthly time step. It was driven by monthly meteorological data. Field data gathered on KBU site in 2003 and annual and monthly biomass, soil water content, and evapotranspiration were validated with the measured data, showing high consistences each other.

Future estimations of biomass and productivity showed that this site was sensitive under environmental change conditions, especially more sensitive under precipitation change than temperature change. Considering precipitation, temperature, and CO_2 fertilization effect, NPP of this ecosystem will decrease up to around 68% after 100 years under a similar climatic change undergone until now even non-grazing condition.

Acknowledgements

The author expresses special thanks to his supervisor, Professor Takehisa Oikawa, Institute of Biological Sciences, University of Tsukuba. The special thanks are extended to the author's elder brother, Lee Gilzae, in the doctoral course of Tsukuba University. The author wishes to express his thanks to Dr. Li Shengon, Tsukuba University for valuable comments and data.

This study has been financially supported by the CREST project (The Rangelands Atmosphere – Hydrosphere – Biosphere Interaction Study Experiment in Northeastern Asia) of JST (Japan Science and Technology Agency).

Reference

Amthor, J.S., 1989. Respiration and Crop Productivity. Springer, New York, p. 215.

Archer, S., 1996. Assessing and Interpreting Grass-Woody Plant Dynamics, in Hodgson, J. and Illius, A.W. (eds.). The Ecology and Management of Grazing Systems, CAB International, Oxford, pp. 101–134.

Baatarbileg, N., Pederson, N., Jacoby, G., D'Arrigo, R., Dugarjav, C., & Mijiddory, R., 2001. An extended drought and streamflow variability record: Potential forest/steppe implications. Open Symposium on Climate and Sustainability of Pastoral Land Use Systems in Temperate and Central Asia. Ulaanbaatar, Mongolian, June 28-July 1, 2001.

Bachelet, D., 1989. A simulation model of intraseasonal carbon and nitrogen dynamics of blue grama swards as influenced by above-and belowground grazing. Ecol. Model. 44, 231–252.

Begzsuren S., Ellis J.E., Ojima D.S., Coughenour M.B. and Chuluun T., 2004. Livestock responses to droughts and severe winter weather in the Gobi Three Beauty National Park, Mongolia. Journal of Arid Environments 59, 785-796.

Chase, T. N., Pielke, R. A., Knaff, J., Kittel, T., & Eastman, J., 2000. A comparison of regional trends in 1979 - 1997 depth-averaged tropospheric temperatures. International Journal of Climatology, 20, 503-518.

Christensen Lindsey, Michael B. Coughenour, James E. Lllis and Zuo Zhong Chen, 2004.

Vulnerability of the Asian Typical Steppe to Grazing and Climate Change. Climatic Change 63, 351-368.

Cnoant R.T. and Paustian K. 2002. Potential soil carbon sequestration in overgrazed grassland ecosystems. Global Biogeochemical Cycles 16, 1143-1151.

Coughenour, M. B. and Chen, D. X., 1997. Assessment of Grassland Ecosystem Responses to Atmospheric Change Using Linked Plant-Soil Process Models. Ecol. Appl. 7, 802–827.

Coughenour, M. B. and Parton,W. J., 1997. Integrated Models of Ecosystem Function: A Grassland Case Study, in Walker, B. and Steffen, W. (eds.), Global Change and Terrestrial Ecosystems, International Geosphere-Biosphere Programme Book Series, Cambridge University Press, Cambridge, pp. 93–114.

Detling, J.K., Parton, W.J., Hunt, H.W., 1979. A simulation model of Bouteloua gracilis biomass dynamics on the North American shortgrass prairie. Oecologia 38, 167–191.

Di castri, F., 1998. A new look at ecotones: emerging international projects on landscape boundaries. Biol. Inst. Sqec. Iss, 17.

Fernandez-Gimenez M. E. and Allen-Diaz B. 1999. Testing a non-equilibrium model of rangeland vegetation dynamics in Mongolia. Journal of Applied Ecology 36, 871-885.

Gao, Z. and Zhang, X., 1997. A Simulation Study of Responses of the Northeast China Transect to Elevated CO2 and Climate Change. Ecological Applications 7, 470–483.

Hanson, J. D., Skiles, J. W., and Parton, W. J., 1988. A Multi-Species Model for Rangeland Plant Communities. Ecol. Model. 44, 89–123.

Hunt, H. W., Trlica, M. J., Redente, E. F., Moore, J. E., Detling, J. K., Kittel, T. G. F., Walter, D. E., Fowler,M. C., Klein, D. A., and Elliott, E. T., 1991. Simulation Model for the Effects of Climate Change on Temperate Grassland Ecosystems. Ecol. Model. 53, 205–246.

Ito, A. and T. Oikawa, 2002. A simulation model of the carbon cycle in land ecosystems (Sim-CYCLE): A description based on dry-matter production theory and plot-scale validation, Ecol. Model., 151, 147-179.

Kuroiwa, S., 1966. Dry matter production of plants, Ecology and Evolution. A Series of Modern Biology, Iwanami Shoten, Tokyo, pp. 71–100, (in Japanese)

Lauenroth, W.K., 1979. Grassland primary production: North American grasslands in perspective. In: French, N.L. (Ed.), Perspectives in Grassland Ecology. Springer, Berlin/Heidelberg/New York, pp. 2–24.

Le Houérou, H.N., 1984. Rain use efficiency: a unifying concept in arid-land ecology. J. Arid Environ. 7, 213–247.

Li S.-G., Asanuma J., Eugster W., Kotani A., Liu J.-J., Urano T., Oikawa T., Davaa G., Oyunbaatar D., and Sugita M., 2005. Net ecosystem carbon dioxide exchange over grazed steppe in central Mongolia. Global Change Biology 11, 1941-1955.

Liang, Y.M., Hazlett, D.L., Lauenroth, W.K., 1989. Biomass dynamics and water use efficiencies of five plant communities in the shortgrass steppe. Oecologia 80, 148–153.

Lindsey Christensen, Michael B. Coughenour, James E. Ellis and Zuo Zhong Chen, 2004. Vulnerability of the Asian typical steppe to grazing and climate change. Climatic Change 63, 351-368.

Liu J.J., 2004. Influence of grazing pressures on belowground biomass and productivity in Mongolia steppe. Japan Earth and Planetary Science Joint Meeting.

Ludwig, J. A., Coughenour, M. B., Liedloff, A. C., and Dyer, R., 2001. Modelling the Resilience of Australian Savanna Systems to Grazing Impacts. Environment International 27, 167–172.

Manabe, S., 1969. Climate and the ocean circulation. I. The atmospheric circulation and the hydrology of the earth's surface. Mon. Weather Rev. 97, 739–774.

Mariko Shigeru, Urano tadaaki and Oikawa Takehisa, 2003. Biomass and carbon fluxes in a Mongolian grassland. The second workshop on terrestrial change in Mongolia. Ulaanbaatar, Mongolia.

McCarthy J, Canziani O, Leary N, Dokken D, White K, 2001. Climate change 2001: Impacts, adaptation, and vulnerability. Cambridge University Press, New York, 1000pp.

McCree, K.J., 1970. An equation for the rate of respiration of white clover plants grown under

controlled conditions. In: Setlik, I. (Ed.), Prediction and Measurement of Photosynthetic Productivity. Proc. IBP/PP Tech. Meet., Trebon. PUBOC, Wageningen, The Netherlands, pp. 221-229.

Milchunas, D.G., Lauenroth, W.K., 1992. Carbon dynamics and estimates of primary production by harvest, 14C dilution, and 14C turnover. Ecology 73 (2), 593–607.

Mongolian Action Program, 2000. National Agency for Meteorology, Hydrology and Environmental Monitoring of Mongolia. Ulaanbaatar, Mongolia, JEMR Press.

Monsi, M., Saeki, T., 1953. Űber den Lichtfaktor in den Pfanzengesellschaften und seine Bedeutung für die Stoffproduktion. Jpn. J. Bot. 14, 22–52.

Natsagdorj, L., 2000. Climate change. In P. Batima, & D. Dagvadorj (Eds.), Climate change and its impacts on Mongolia. Ulaanbaatar, Mongolia, JEMR Press.

Neilson, R. P., Prentice, I. C., Smith, B., Kittel, T., and Viner. D., 1998, Simulated Changes in Vegetation Distribution under Global Arming, in Watson, R. T., Zinyoweie, M. C., and Moss, R. H.(eds.). The Regional Impacts of Climate Change: An Assessment of Vulnerability, Cambridge University Press, N.Y., pp. 439–456.

Nosberger, J., Blum, H., and Fuhrer, J., 2000. Crop Ecosystem Responses to Climatic Change: Productive Grasslands, in Reddy, K. R. and Hodges, H. F. (eds.). Climate Change and Global Crop Productivity, CABI Publishing, N.Y., pp. 271–291.

Nouvellon Y., S. Rambal, D. Lo Seen, M.S. Moran, J.P. Lhomme, A. Bégué, A.G. Chehbouni, Y. Kerr, 2000. Modelling of daily fluxes of water and carbon from shortgrass steppes. Agricultural and Forest Meteorology 100, 137–153.

Parton, W. J., Schimel, D. S., Cole, C. V., and Ojima, D. S., 1987, Analysis of Factors Controlling Soil Organic Matter Levels in Great Plains Grasslands, Soil Science Society of America Journal 51, 1173–1197.

Polley, H. W., Morgan, J. A., Cambell, B. D., and Smith, M. S., 2000, Crop Ecosystem Responses to Climatic Change: Rangelands, in Reddy, K. R. and Hodges, H. F. (eds.). Climate Change and Global Crop Productivity, CAB International, N.Y., pp. 293–314.

Ruimy, A., Delien, G., Saugier, B., 1996. TURC: a diagnostic model of continental gross primary productivity and net primary productivity. Glob. Biogeochem. Cycles 10, 269-286.

Sala, O.E., Parton, W.J., Joyce, L.A., Lauenroth, W.K., 1988. Primary production of the central grassland region of the United States. Ecology 69 (1), 40–45.

Seastedt TR, Hayden BP, Owensby CE, Knapp AK, 1998. Climate change, elevated CO_2 and predictive modeling: Past and future climate change scenarios for the tallgrass Prairie. In: Grassland Dynamics: Long-Term Ecological Research in Tallgrass Prairie (eds Knapp, AK, Briggs, JM, Hartnett, DC, Collins, SC), pp. 283–300. Oxford Press, New York.

Seligman N.G. and Van Leulen N, 1989. Herbage production of a Mediterranean grassland in

relation to soil depth, rainfall and nitrogen: a simulation study. Ecol. Model. 47, 303-311

Sims, P.L., Singh, J.S., 1978. The structure and the function of ten western north American grasslands. III. Net primary production. Turnover and efficiencies of energy capture and water use. J. Ecol. 66, 573–597.

Smith JB, Bhatti N, Menzhulin G, Benioff R, Campos M, Jallow B, Rijsberman F, Budyko Mi, Dixon RK, 1996. Adapting to climate change: Assessment and Issues. Springer-Verlag, New York, 475pp.

Olson J. S., Watts J. A., and Allison L. J., 1983. Carbon in live vegetation of major world ecosystems. ORNL-5862. Oak Ridge National Laboratory, Oak Ridge.

Urano tadaaki, Mariko Shigeru, Kiyokaza jawada, 2004. Seasonal dynamics of biomass and carbon fluxes in a Mongolian grassland. Japan Earth and Planetary Science Joint Meeting.

Wand, S. J. E., Midgley, G. F., Jones, M. H., and Curtis, P. S., 1999. Responses of Wild C4 and C3 Grass (Poaceae) Species to Elevated Atmospheric CO2 Concentration: A Meta-Analytic Test of Current Theories and Perceptions, Global Change Biology 5, 723–741.

Wilhite DA, 1993 Drought Assessment, Management and Planning: Theory and Case Studies. Kluwer Academic Publishers, Hingham, 38 MA, 293pp.

Webb, W., Szarek, S., Lauenroth, W., Kinerson, R., Smith, M., 1978. Primary productivity and water use in native forest, grassland, and desert ecosystems. Ecology 59 (6), 1239–1247.

World Resources Institute (WRI), 2003. A Guide to World Resources 2002-2004: Decisions for the Earth Balance, Voice, and Power. 43 World Resources Inst., Washinton DC., 328pp.

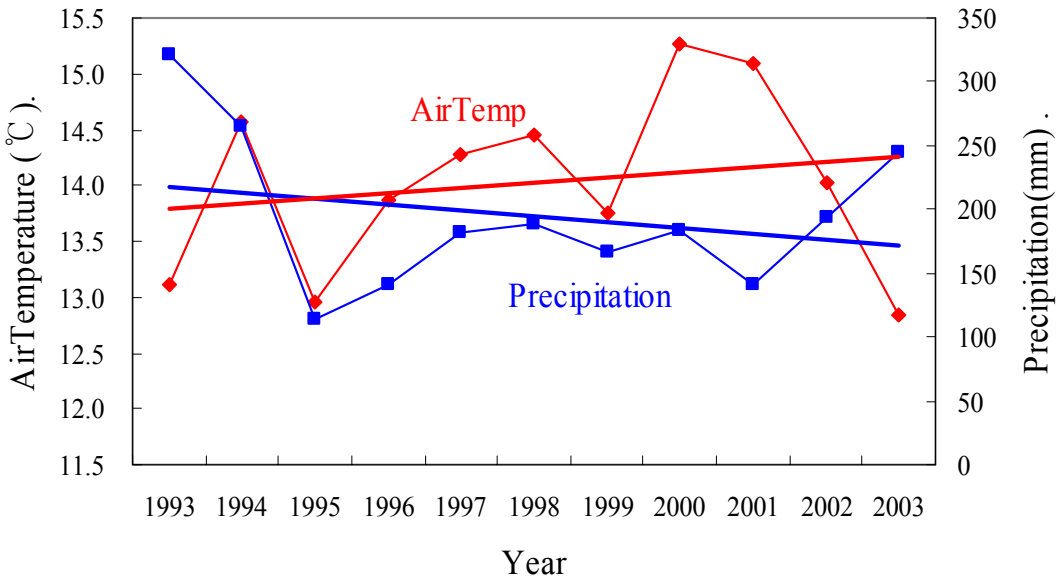


Fig. 1 Annual mean air temperature and precipitation in KBU during 1993-2003 from a local meteorological station of the Institute of Meteorology and Hydrology of Mongolia

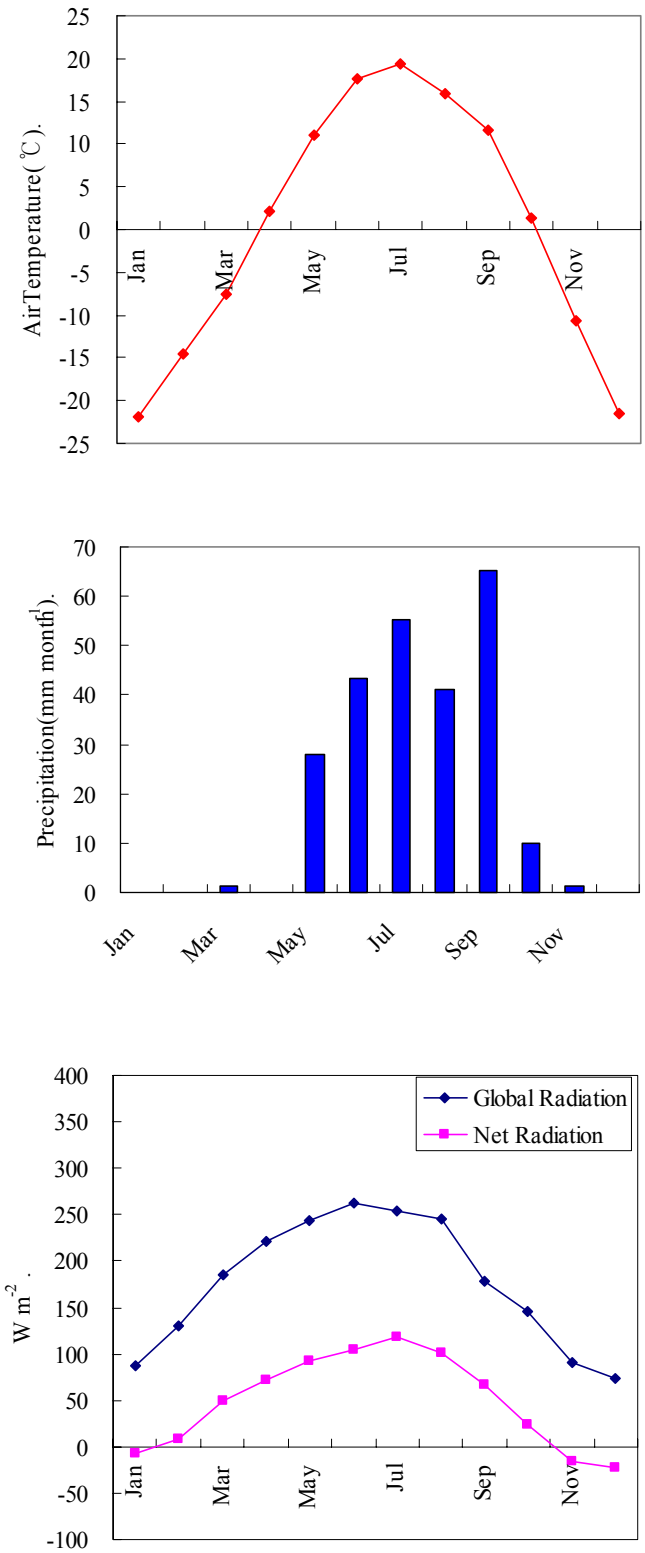


Fig. 2 Meteorological data of KBU at 2003 from RAISE

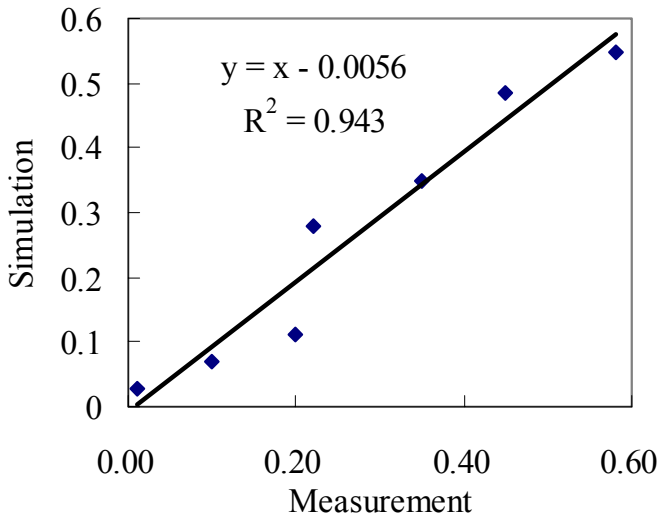
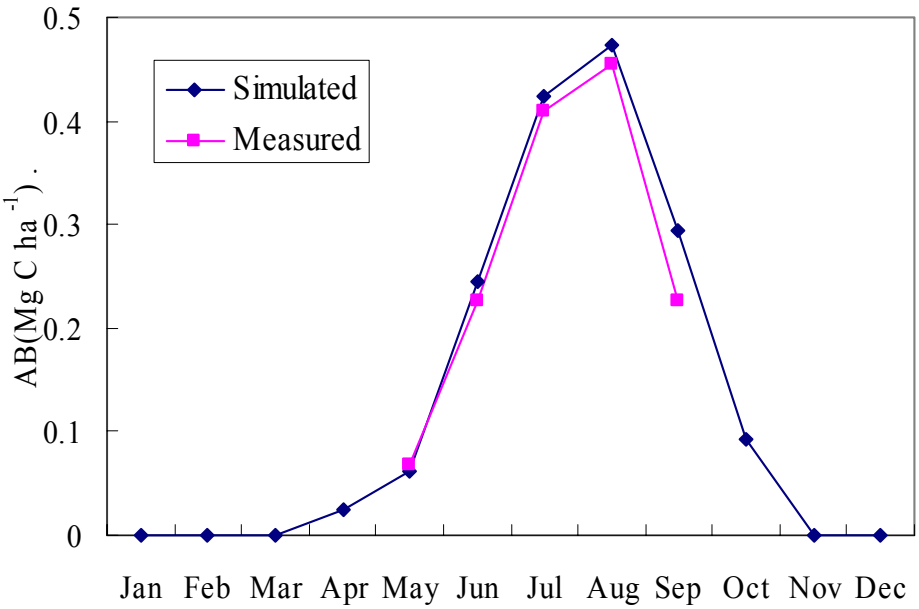


Fig. 3 Linear regression analysis of measured and simulated LAI

a



b

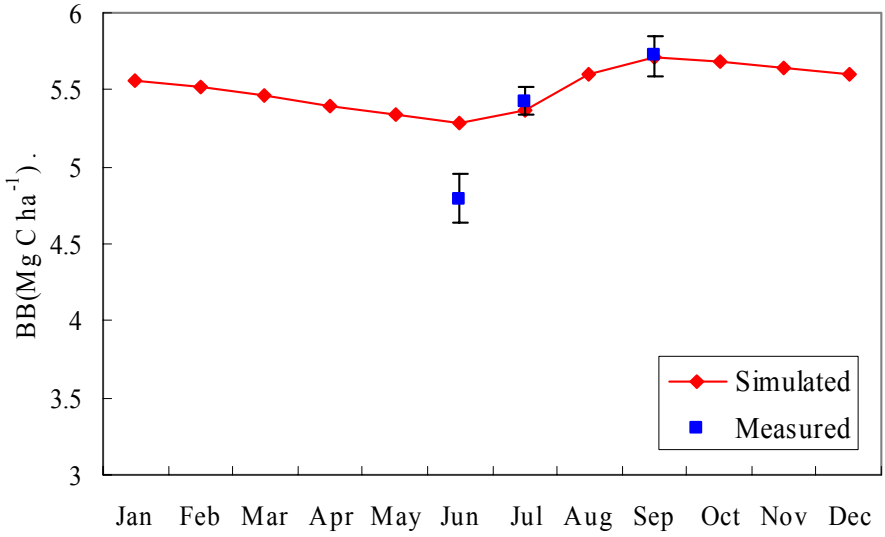
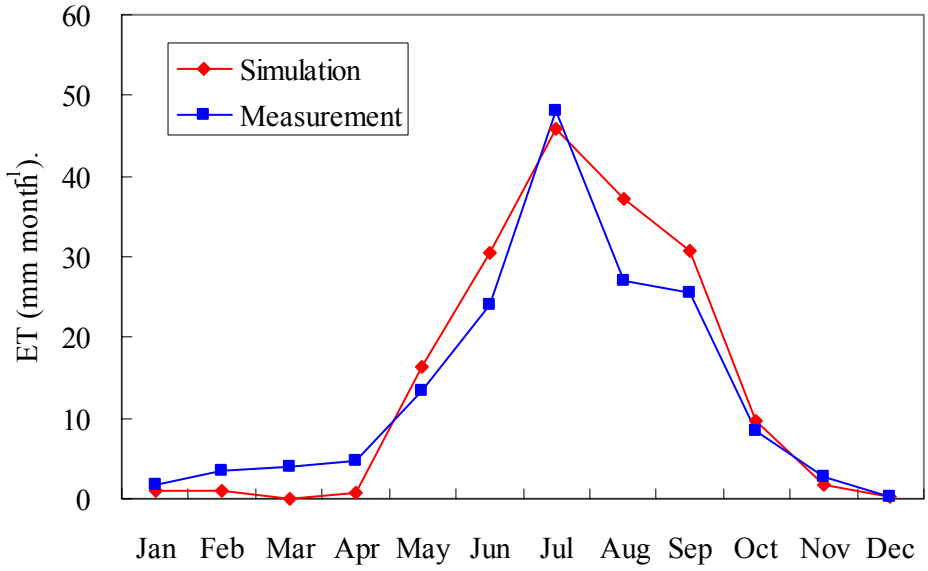


Fig. 4 Monthly change of measured and simulated aboveground biomass (a) and belowground biomass (b)

a



b

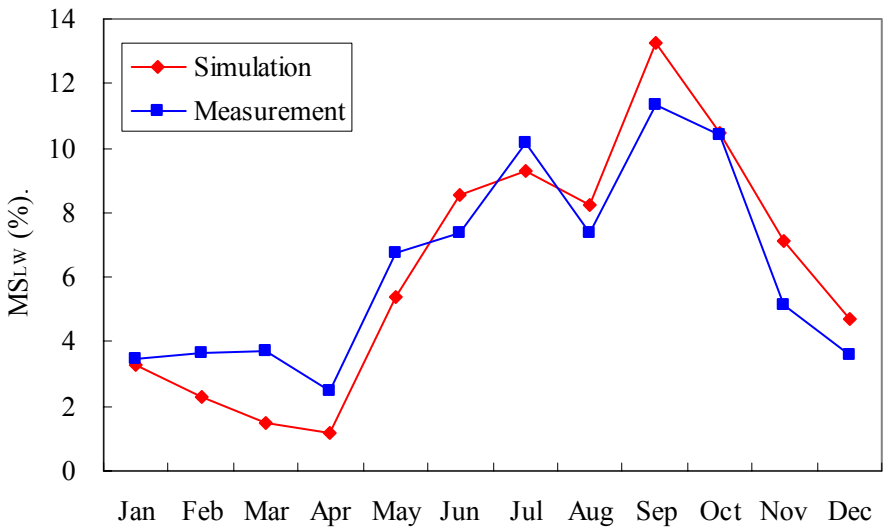


Fig. 5 Monthly change of measured and simulated evapotranspiration (a) and MS_{LW} (b)

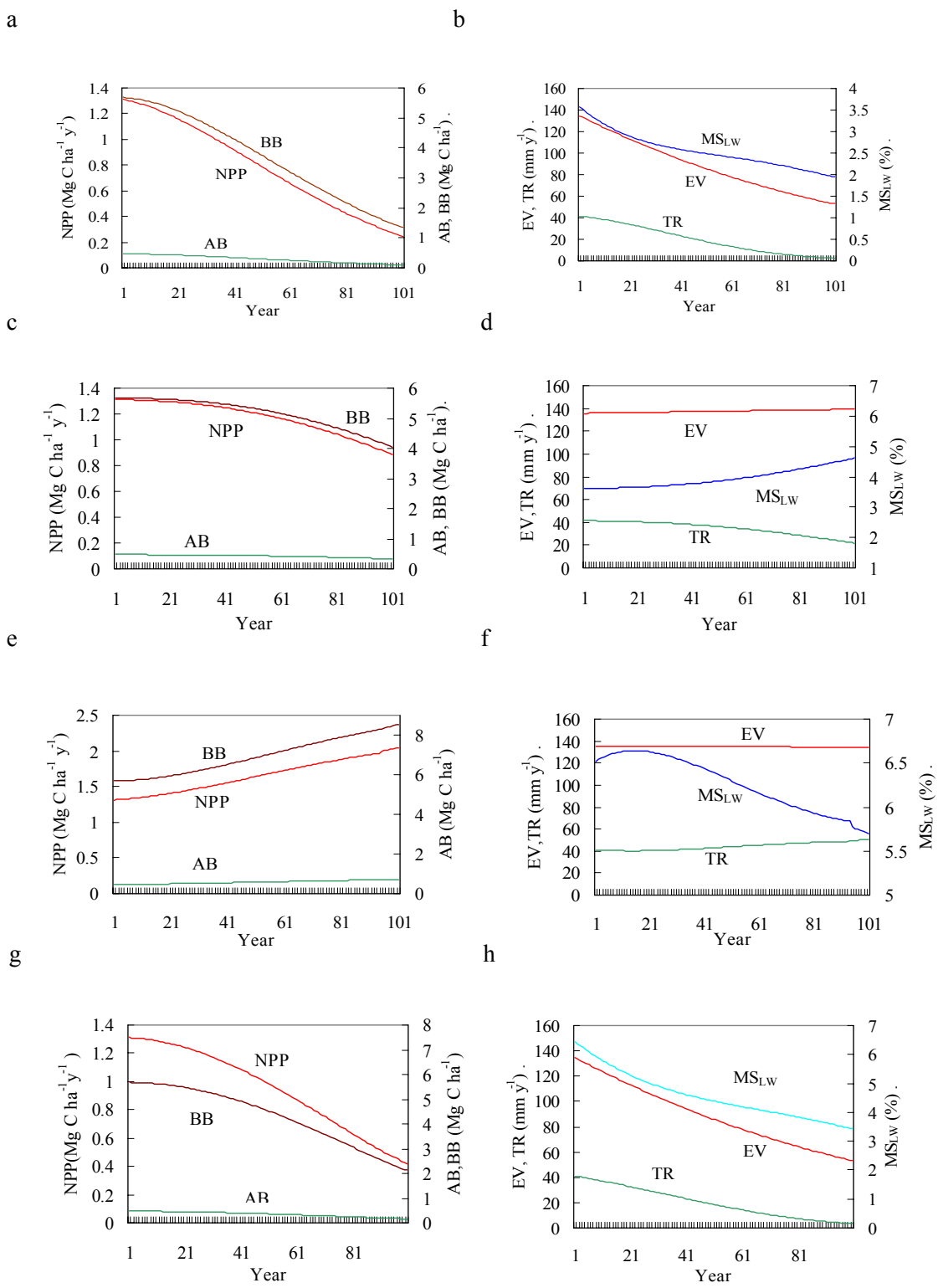


Fig. 6 Results of future carbon/water cycles in S1P (a, b), S2T (c, d), S3C (e, f), and S4A (g, h) during 100 years under non-grazing condition

Table 1

Air temperature from 1993-2003 at KBU from a local meteorological station of the Institute of Meteorology and Hydrology of Mongolia

Month	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	Aver.
Jan	−20.1	−21.0	−21.5	−24.5	−22.8	−23.3	−24.0	−26.0	−27.4	−19.2	−21.82	−22.9
Feb	−12.3	*−16.1*	−14.2	−18.2	−17.0	−11.2	−16.0	−19.7	−21.8	−14.5	−14.45	−16.0
Mar	*−6.6*	−2.6	*−6.6*	−7.1	*−6.6*	−4.2	−11.2	−4.6	−7.4	*−6.6*	−7.52	−6.5
Apr	5.6	6.1	*4.8*	3.8	5.2	*4.8*	5.1	4.8	4.5	3.7	2.14	4.6
May	12.2	13.7	10.2	14.0	14.0	12.8	*13.3*	16.3	13.3	12.9	11.03	13.1
Jun	18.1	20.5	19.4	17.4	20.0	*19.5*	17.4	23.5	21.6	19.1	17.58	19.5
Jul	19.0	20.6	20.2	21.4	21.2	22.0	22.3	22.6	22.1	22.4	19.44	21.2
Aug	16.0	18.7	18.1	*18.3*	18.1	17.5	18.2	18.2	20.6	20.6	15.90	18.2
Sep	11.4	12.2	7.3	11.9	10.4	12.5	10.7	13.3	12.0	11.6	11.66	11.3
Oct	2.0	1.7	2.6	0.3	2.0	2.6	0.6	−2.3	*0.9*	−2.5	1.41	0.9
Nov	−11.5	−6.5	−6.3	−15.3	−10.1	*−10.0*	−9.2	*−10.0*	−7.8	*−10.0*	−10.73	−9.8
Dec	−17.3	−20.1	−16.5	−19.0	−19.1	−21.6	*−20.4*	−22.9	−23.1	−23.5	−21.46	−20.5
Aver.	1.4	2.3	1.4	0.2	1.3	1.8	0.6	1.1	0.6	1.2	0.3	1.1

* Italic numbers are average values during 1993-2003 because of the absent of measurement data.

Table 2

Surface temperature from 1993-2003 at KBU from a local meteorological station of the Institute of Meteorology and Hydrology of Mongolia

Month	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	Aver.
Jan	−20.5	−21.3	*−24.8*	−26.5	−25.8	−26.3	−28.7	−29.4	−30.7	−20.9	−17.87	−24.8
Feb	−12.2	*−18.3*	*−18.3*	−21.5	−21.0	−11.4	−24.4	−22.1	−24.6	−15.8	−11.58	−18.3
Mar	*−5.3*	−0.1	*−5.3*	−5.1	*−5.3*	−2.3	−11.8	−4.4	−8.3	*−5.3*	−5.23	−5.3
Apr	10.6	10.5	*8.2*	8.0	7.6	*8.2*	8.2	9.1	7.3	7.5	5.15	8.2
May	18.0	20.0	15.4	19.3	20.0	18.1	*18.6*	22.8	19.2	18.4	14.66	18.6
Jun	23.6	25.7	25.0	27.7	27.9	*26.1*	24.3	32.0	28.2	25.3	21.36	26.1
Jul	22.1	26.0	26.3	28.2	27.4	30.4	29.4	29.8	29.6	29.9	22.78	27.5
Aug	19.1	22.9	24.3	*23.3*	22.8	22.2	24.5	22.7	26.8	27.8	19.71	23.3
Sep	14.3	14.4	9.6	15.6	13.7	15.2	14.6	17.4	14.5	15.8	14.15	14.5
Oct	3.9	*2.6*	3.3	1.3	3.0	3.3	0.9	−0.8	*2.6*	−0.7	2.46	2.0
Nov	−12.8	*−11.3*	−6.9	−19.4	−10.5	*−11.3*	−11.0	*−11.3*	−7.9	*−11.3*	−10.36	−11.3
Dec	−19.7	*−22.6*	−18.3	−23.6	−21.2	−25.1	*−22.6*	−26.1	−25.5	−25.4	−18.29	−22.6
Aver.	3.4	4.0	3.2	2.3	3.2	3.9	1.8	3.3	2.6	3.8	3.1	3.2

* Italic numbers are average values during 1993-2003 because of the absent of measurement data.

Table 3

Precipitation from 1993-2003 at KBU from a local meteorological station of the Institute of Meteorology and Hydrology of Mongolia

Month	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	Aver.
Jan	0.3	2.9	0.0	4.3	0.0	2.8	0.0	1.9	5.9	3.2	0.0	1.9
Feb	1.3	*1.3*	0.0	0.0	4.4	0.2	0.0	1.1	0.9	1.9	0.0	1.0
Mar	*1.1*	0.0	*1.1*	1.4	*1.1*	0.0	1.3	1.2	1.6	*1.1*	1.2	1.0
Apr	4.9	1.4	*1.7*	0.4	0.0	*1.7*	0.8	2.0	2.6	2.9	0.0	1.7
May	8.7	6.9	1.2	3.7	1.9	3.2	*6.8*	0.0	4.0	31.3	28.0	8.7
Jun	25.4	18.0	22.6	17.7	20.7	*30.1*	62.3	8.4	22.6	72.9	43.2	31.3
Jul	89.9	140.3	18.5	94.7	74.8	22.8	42.6	66.6	33.0	41.1	55.2	61.8
Aug	152.2	67.9	59.6	*65.3*	59.8	126.7	14.4	76.7	27.6	27.4	41.0	65.3
Sep	16.8	20.2	7.4	4.0	15.5	22.0	33.4	8.6	31.2	1.3	65.1	20.5
Oct	9.8	1.3	3.0	6.3	0.0	11.6	0.0	11.8	*5.0*	6.2	10.1	5.9
Nov	6.2	0.0	0.0	6.4	2.2	*3.2*	2.7	*3.2*	2.2	3.2	1.2	2.8
Dec	5.4	5.6	2.2	1.3	1.8	0.0	*2.6*	1.2	4.8	1.4	0.0	2.4
Aver.	320.9	264.5	114.5	140.2	181.1	189.3	166.9	182.7	141.4	193.9	245.0	204.3

* Italic numbers are average values during 1993-2003 because of the absent of measurement data.

Table 4

List of parameters used in the model. Other parameters were used with same values in Sim-CYCLE.

Term	Equation	This study	Sim-CYCLE	Unit
PC_{SAT0}	EQ. (7)	24.7	14.0	micro mol CO_2 m^{-2} s^{-1}
$SARG_F$	Eq. (10)	0.25	0.65	g C (g C)$^{-1}$
$SARG_C$	Eq. (10)	0.25	0.20	g C (g C)$^{-1}$
$SARG_R$	Eq. (10)	0.20	0.20	g C (g C)$^{-1}$
$SARM_F$	Eq. (9)	1.75	2.45	mg C (g C day)$^{-1}$
$SARM_C$ (heart)	Eq. (9)	0.11	1.30	mg C (g C day)$^{-1}$
$SARM_R$ (heart)	Eq. (9)	0.11	1.02	mg C (g C day)$^{-1}$
SLF_F ($\times 10^{-3}$)	Eq. (12)	6.96	4.95	
SLF_C ($\times 10^{-3}$)	Eq. (12)	0.71	0.71	
SLF_R ($\times 10^{-3}$)	Eq. (12)	0.26	1.22	
T_{OPT}		22	20	℃
T_{MIN}		0	0	℃
T_{MAX}		45	40	℃

PC_{SAT0}, potential maximum of light-saturated photosynthetic rate; $SARG_{F,C,R}$, specific growth respiration; $SARM_{F,C,R}$, specific maintenance respiration rate; $SLF_{F,C,R}$, specific litter fall rate; $T_{OPT, MIN, MAX}$, optimum, minimum, and maximum temperature for photosynthesis

Table 5

Comparison between the measurements and simulated results at KBU

	Units	Measurement	Simulation	Relative Error (%)
AB	Mg C ha^{-1}	0.47	0.47	1.0
BB	Mg C ha^{-1}	5.72	5.72	0.1
ANPP	Mg C ha^{-1} $year^{-1}$	0.45	0.45	2.0
BNPP	Mg C ha^{-1} $year^{-1}$	0.85	0.86	1.0
HR	Mg C ha^{-1} $year^{-1}$	1.23	1.23	0.2
LAI^a	m^2/m^2	0.58	0.55	5.5
MS_{LW}	%	6.3	6.3	0.3
ET	mm $year^{-1}$	163	175	6.8

Relative Error (%) = |Measurement – Simulation|/Simulation × 100

AB, aboveground biomass; BB, belowground biomass; ANPP, aboveground net primary productivity; BNPP, belowground net primary productivity; HR, heterotrophic respiration; MS_{LW}, lower soil moisture content; ET, evapotranspiration; LAI^a, Maximum leaf area index.

Table 6

Terms of the energy, water, and carbon budgets for 2003

	Units	Simulation
GPP	Mg C ha^{-1} $year^{-1}$	2.33
EP	Mg C ha^{-1} $year^{-1}$	2.08
NPP	Mg C ha^{-1} $year^{-1}$	1.31
AR	Mg C ha^{-1} $year^{-1}$	1.01
ARM	Mg C ha^{-1} $year^{-1}$	0.24
ARG	Mg C ha^{-1} $year^{-1}$	0.77
Aboveground respiration	Mg C ha^{-1} $year^{-1}$	0.53
Belowground respiration	Mg C ha^{-1} $year^{-1}$	0.48
NEP	Mg C ha^{-1} $year^{-1}$	0.16
Allocation to roots	Mg C ha^{-1} $year^{-1}$	1.22
EV	mm $year^{-1}$	133
TR	mm $year^{-1}$	42
Global radiation	MJ m^{-2} $year^{-1}$	5574^{M}
Net radiation	MJ m^{-2} $year^{-1}$	1563^{M}
Precipitation	mm $year^{-1}$	245^{M}

GPP, gross primary productivity; EP, effective photosynthate for biomass growth [=GPP - ARM]; NPP, net primary productivity; AR, autotrophic respiration; ARM, autotrophic maintenance respiration; ARG, autotrophic growth respiration, NEP, net ecosystem productivity; EV, evaporation rate; TR, transpiration rate.

^{M}: measurement data.

Table 7

Results of future carbon/water cycles after 100 years

	Carbon Cycle			Water Cycle		
	NPP	AB	BB	MS_{LW}	EV	TR
S1P	-81	-79	-76	-45	-61	-94
S2T	-32	-31	-28	27	3	-47
S3C	55	53	50	-12	-1	22
S4A	-68	-66	-63	-47	-61	-92

Unit: % relative to values of the first year

Appendix:

Program list of the improved Sim-CYCLE in Mongolian grassland

The improved Sim-CYCLE is coded in MFC (Microsoft Foundation Classes) which is a Microsoft library that wraps portions of the Windows API in C^{++} classes, forming an application framework. This language is object oriented language, so that the code is easy to be understood. The model is executable under Windows platform. The code is divided into 16 class files (each class file has a header and a cpp file) and 1 global header file.

Global header file:	prototype.h	(prototype declaration)
Class files:	Bmas	(ecosystem carbon storage)
	Cflx	(ecosystem carbon fluxes)
	Echar	(ecosystem characteristics)
	Gchar	(grid characteristics)
	Gcon	(grid condition)
	Pchar	(vegetation characteristics)
	Pflx	(plant carbon fluxes)
	Pmas	(plant biomass)
	PrintData	(print calculated data)
	Schar	(soil characteristics)
	Sflx	(soil carbon fluxes)
	Smas	(soil carbon storage)
	TransSimulation	(future simulation)
	Initiate	(initiate variables)
	SimCalculation	(calculation of carbon/water process)
	SimCycleMain	(main function of Sim-CYCLE)

In order for simulation, insert OnMainBut() for point simulation and OnTransSimul() for future simulation in a Dialog Box after making the dialog box (file name: CSimCycleMFC_OPDlg). These files should be compiled and linked into an executable file.

```
/*          Simulation model of Caron cYCle in Land Ecosystems                  */
/*          Developed by A. Ito in University of Tsukuba at Aug. 2000 with C language   */
/*          P.Z. Lee improved the Sim-CYCLE with MFC for simulating Mongolian grassland  */

void CSimCycleMFC_OPDlg::OnMainBut()
{
        m_cSCM = new SimCycleMain();
        m_cSCM->OnePointSimulation();
        MessageBox("Work has just been finished");
}
void CSimCycleMFC_OPDlg::OnTransSimul()
{
        UpdateData(true);
        TransSimulation *pTrans = new TransSimulation();
        pTrans->SetFile("data/trans/");
        // condition of precipitation rate inputed by Dialog Box
        pTrans->m_dPreRate = m_dPreRate * 0.01;
        // condition of temperature inputed by Dialog Box
        pTrans->m_dTempRate = m_dTempRate;
        TransSimulation *pStable = new TransSimulation();
        pStable->SetFile("data/stable/");
        m_cSCM = new SimCycleMain();
        pTrans->m_iTSimYear = m_iTransRound; // 500 years simulation
        m_cSCM->OPTSimulation(pTrans, pStable);
        pTrans->FileClose(); pStable->FileClose();
        delete pTrans;delete pStable;
        MessageBox("Work has just finished");
}
```

prototype.h

```
/* prototype declaration */
#define dTr 0.0174533                    /* angle conversion, from degree to radian */
#define rTd 57.29577951                  /* angle conversion, from radian to degree */
#define PI 3.141592653                   /** pai **/
#define cdTc 0.272727                    /* from CO2-base to Crabon-base */
#define dmTc 2.2                         /* from dry-matter-base to Carbon base */
#define cTdm 0.4545                      /* from dry-matter-base to Carbon base */
#define lTs (3600.0*12.0/100000000.0)    /* from micro-mol m-2 s-1 to Mg C ha-1 day-1*/
#define ZAT 273.15                       /* zero degree centigrade in absolute temperature */

#define INIT_MASS 0.1
#define TERM_HYD 0.1
#define TER_CON 0.001    /* criteria for determining the equilibrium, NEP value in Mg C ha-1 yr-1 */

/*****      PHILLY APPEND      *****/
#define g_iCO2YEAR 2003
#define STATIC 1
static int MONDAY[12] = {31,28,31,30,31,30,31,31,30,31,30,31};

// variables
#define FILENUM 100

static char R3DIR[] = "data/result/";
static char GOGEDIR[] = "data/source/";
static char ANADIR[] = "data/ans/";
static CString DIR_HAEDER = "F:/Graduate_Tsukuba/project/ProjectMongo/Data/ProgramData/";
```

```
// Bmas.h: interface for the Bmas class.
#if !defined(AFX_BMAS_H__60473CBA_01D6_4F7C_AFC0_B156BAB21D51__INCLUDED_)
#define AFX_BMAS_H__60473CBA_01D6_4F7C_AFC0_B156BAB21D51__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
#include "Pmas.h"
#include "Smas.h"
class Bmas
{
public:
		Pmas	*c3; /*	C3	plant	mass	*/
		Pmas	*c4; /*	C4 plant mass	*/
		Pmas	*plant; /*	all	plant	mass	*/
		Smas	*soil; /*	soil	mass	*/
		double		*total; /*	ecosystem	total	carbon	storage, Mg C ha-1	*/
public:
		Bmas();virtual ~Bmas();
};
#endif

// Bmas.cpp: implementation of the Bmas class.
#include "stdafx.h"
#include "SimCycleMFC_OP.h"
#include "Bmas.h"
#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif
Bmas::Bmas()
{
		c3 = new Pmas();c4 = new Pmas();
		plant = new Pmas();soil = new Smas();
		total = (double*)calloc(DAYNUM, sizeof(double));
}
Bmas::~Bmas()
{
		free(total);	delete c3;	delete c4;	delete plant;delete soil;
}
```

```
// flx.h: interface for the Cflx class.
#if !defined(AFX_FLX_H__FF09D745_6CED_487D_92A8_05E03FDB3F08__INCLUDED_)
#define AFX_FLX_H__FF09D745_6CED_487D_92A8_05E03FDB3F08__INCLUDED_
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
#include "Pflx.h"
#include "Sflx.h"
class Cflx
{
public:
		Pflx	*c3; /*	C3 plant fluxes	*/
		Pflx	*c4; /*	C4 plant fluxes	*/
		Pflx	*plant; /*	total plant fluxes	*/
		Sflx	*soil; /*	soil fluxes	*/
		double		npp_miami;	/*	NPP	estimated	by	Miami	model, Mg C ha-1 yr-1*/
		double		npp_montreal;	/*	NPP	estimated	by	Montreal	model, Mg C ha-1 yr-1*/
		double		npp_rosenzweig;	/*	NPP	estimated	by	Montreal	model, Mg C ha-1 yr-1*/
		double		npp_chikugo;	/*	NPP	estimated	by	Chikugo	model, Mg C ha-1 yr-1*/
		double		npp_a_chikugo;	/*	NPP	estimated	by	Chikugo	model, Mg C ha-1 yr-1*/
		double		*nep;	/*	net	ecosystem production,	Mg C ha-1 mon-1	*/
		double		*ncb;	/*	net	carbon	balance	of	grid,	Mg C ha-1 mon-1 */
public:
		void plant_stand(Gchar *grid, Gcon *loct, Bmas *mass);
		void biome_processes(Gchar *grid, Gcon	*loct, Echar *ecochar,Bmas *mass);
		void Vanish(Bmas *mass);
		void plant_flux_zero(long	month, Pflx *flux);
		void vlzero(Gchar *grid, Pmas *mass, Pflx *flux);
public:
		Cflx();virtual ~Cflx();
};
#endif
// flx.cpp: implementation of the Cflx class.
#include "stdafx.h"
#include "SimCycleMFC_OP.h"
#include "flx.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif
Cflx::Cflx()
{
		c3 = new Pflx();c4 = new Pflx();
		plant = new Pflx();soil = new Sflx();
		nep = (double*)calloc(DAYNUM, sizeof(double));
		ncb = (double*)calloc(DAYNUM, sizeof(double));
}
Cflx::~Cflx()
{		free(nep);free(ncb);delete c3;delete c4;delete plant;delete soil;}
void Cflx::vlzero(Gchar *grid, Pmas *mass, Pflx *flux)
{
		mass->fol=0.0;mass->stm=0.0;mass->rot=0.0;

		mass->mfol[grid->month]=mass->lai[grid->month]=0.0;
		mass->mstm[grid->month]=0.0;mass->mrot[grid->month]=0.0;mass->plant[grid->month]=0.0;
		plant_flux_zero(grid->month, flux);
}

void Cflx::plant_flux_zero(long month, Pflx *flux)
{
		flux->gpp[month]=0.0;	flux->epp[month]=0.0;
```

```
		flux->spp[month]=0.0;	flux->npp[month]=0.0;

		flux->fnpp[month]=0.0;	flux->cnpp[month]=0.0;	flux->rnpp[month]=0.0;

		flux->rfm[month]=0.0;	flux->rcm[month]=0.0;	flux->rrm[month]=0.0;
		flux->rpm[month]=0.0; flux->rfg[month]=0.0;	flux->rcg[month]=0.0;
		flux->rrg[month]=0.0;	flux->rpg[month]=0.0;	flux->rp[month]=0.0;

		flux->lf[month]=0.0;	flux->lc[month]=0.0;	flux->lr[month]=0.0;	flux->lL[month]=0.0;
		flux->tpp[month]=0.0; flux->tpf[month]=0.0; flux->tpc[month]=0.0; flux->tpr[month]=0.0;
		flux->hvst[month]=0.0;
}
void Cflx::Vanish(Bmas *mass)
{
		long k;
		/*	tentative	mass	values	*/
		(mass->plant)->fol=(mass->c3)->fol=(mass->c4)->fol=0.0;
		(mass->plant)->stm=(mass->c3)->stm=(mass->c4)->stm=0.0;
		(mass->plant)->rot=(mass->c3)->rot=(mass->c4)->rot=0.0;
		(mass->soil)->ltr=0.0;
		(mass->soil)->msl=0.0;
		for(k=0;k<DAYNUM;k++){
			/*	monthly	mass	values	*/
			(mass->c3)->mfol[k]=(mass->c4)->mfol[k]=(mass->plant)->mfol[k]=0.0;
			(mass->c3)->lai[k]=(mass->c4)->lai[k]=(mass->plant)->lai[k]=0.0;
			(mass->c3)->mstm[k]=(mass->c4)->mstm[k]=(mass->plant)->mstm[k]=0.0;
			(mass->c3)->mrot[k]=(mass->c4)->mrot[k]=(mass->plant)->mrot[k]=0.0;
			(mass->c3)->plant[k]=(mass->c4)->plant[k]=(mass->plant)->plant[k]=0.0;
			(mass->soil)->mltr[k]=0.0;(mass->soil)->mmsl[k]=0.0;(mass->soil)->soil[k]=0.0;

			/*	fluxes	*/
			plant_flux_zero(k, this->c3);plant_flux_zero(k, this->c4);plant_flux_zero(k, (plant));
			(soil)->rl[k]=0.0;(soil)->rh[k]=0.0;(soil)->sf[k]=0.0;(soil)->rS[k]=0.0;
			nep[k]=0.0; ncb[k]=0.0;
		}
}
/************	aggregate	C3	and	C4	community	****************/
void Cflx::plant_stand(Gchar *grid,	Gcon *loct, Bmas *mass)
{
		long f;
		f=grid->month;

		/**	mass	**/
		(mass->plant)->fol=(mass->c3)->fol+(mass->c4)->fol;
		(mass->plant)->stm=(mass->c3)->stm+(mass->c4)->stm;
		(mass->plant)->rot=(mass->c3)->rot+(mass->c4)->rot;
		(mass->plant)->mfol[f]=(mass->c3)->mfol[f]+(mass->c4)->mfol[f];
		(mass->plant)->mstm[f]=(mass->c3)->mstm[f]+(mass->c4)->mstm[f];
		(mass->plant)->mrot[f]=(mass->c3)->mrot[f]+(mass->c4)->mrot[f];
		(mass->plant)->lai[f]=(mass->c3)->lai[f]+(mass->c4)->lai[f];

		(mass->plant)->plant[f]=(mass->c3)->plant[f]+(mass->c4)->plant[f];

		/**	production	**/
		(plant)->gpp[f]=this->c3->gpp[f]*loct->C3ptn+this->c4->gpp[f]*loct->C4ptn;
		(plant)->spp[f]=this->c3->spp[f]*loct->C3ptn+this->c4->spp[f]*loct->C4ptn;
		(plant)->epp[f]=this->c3->epp[f]*loct->C3ptn+this->c4->epp[f]*loct->C4ptn;
		(plant)->npp[f]=this->c3->npp[f]*loct->C3ptn+this->c4->npp[f]*loct->C4ptn;
		/**	maintenance	respiration	**/
		(plant)->rfm[f]=this->c3->rfm[f]*loct->C3ptn+this->c4->rfm[f]*loct->C4ptn;
		(plant)->rcm[f]=this->c3->rcm[f]*loct->C3ptn+this->c4->rcm[f]*loct->C4ptn;
		(plant)->rrm[f]=this->c3->rrm[f]*loct->C3ptn+this->c4->rrm[f]*loct->C4ptn;
		(plant)->rpm[f]=(plant)->rfm[f]+(plant)->rcm[f]+(plant)->rrm[f];
		/**	growth	respiration	**/
		(plant)->rfg[f]=this->c3->rfg[f]*loct->C3ptn+this->c4->rfg[f]*loct->C4ptn;
```

```
		(plant)->rcg[f]=this->c3->rcg[f]*loct->C3ptn+this->c4->rcg[f]*loct->C4ptn;
		(plant)->rrg[f]=this->c3->rrg[f]*loct->C3ptn+this->c4->rrg[f]*loct->C4ptn;
		(plant)->rpg[f]=(plant)->rfg[f]+(plant)->rcg[f]+(plant)->rrg[f];
		/**	total	respiration	**/
		(plant)->rp[f]=(plant)->rpm[f]+(plant)->rpg[f];
		/**	allocation	**/
		(plant)->tpf[f]=this->c3->tpf[f]*loct->C3ptn+this->c4->tpf[f]*loct->C4ptn;
		(plant)->tpc[f]=this->c3->tpc[f]*loct->C3ptn+this->c4->tpc[f]*loct->C4ptn;
		(plant)->tpr[f]=this->c3->tpr[f]*loct->C3ptn+this->c4->tpr[f]*loct->C4ptn;
		(plant)->tpp[f]=this->c3->tpp[f]*loct->C3ptn+this->c4->tpp[f]*loct->C4ptn;
		/**	litterfall	**/
		(plant)->lf[f]=this->c3->lf[f]*loct->C3ptn+this->c4->lf[f]*loct->C4ptn;
		(plant)->lc[f]=this->c3->lc[f]*loct->C3ptn+this->c4->lc[f]*loct->C4ptn;
		(plant)->lr[f]=this->c3->lr[f]*loct->C3ptn+this->c4->lr[f]*loct->C4ptn;
		(plant)->lL[f]=(plant)->lf[f]+(plant)->lc[f]+(plant)->lr[f];
		(plant)->fnpp[f] = ((mass->plant)->fol - (mass->plant)->fol_p) + (plant)->lf[f];
		(plant)->cnpp[f] = ((mass->plant)->stm - (mass->plant)->stm_p) + (plant)->lc[f];
		(plant)->rnpp[f] = ((mass->plant)->rot - (mass->plant)->rot_p) + (plant)->lr[f];
		(mass->plant)->fol_p=(mass->plant)->fol;
		(mass->plant)->stm_p=(mass->plant)->stm;
		(mass->plant)->rot_p=(mass->plant)->rot;
		/**	harvest	**/
		(plant)->hvst[f]=this->c3->hvst[f]*loct->C3ptn+this->c4->hvst[f]*loct->C4ptn;
}
/********	Biome-specific	processes	********/
void Cflx::biome_processes(Gchar *grid, Gcon	*loct, Echar *ecochar, Bmas	*mass)
{
		(ecochar->c3)->GrowthPeriod(grid, loct);(ecochar->c4)->GrowthPeriod(grid, loct);
		(ecochar->c3)->grass_process(grid, loct, this->c3, (mass->c3));
		(ecochar->c4)->grass_process(grid, loct, this->c4, (mass->c4));
}
```

```
// Echar.h: interface for the Echar class.
#if !defined(AFX_ECHAR_H__84E53224_3C47_4247_ABE2_2D44193684BB__INCLUDED_)
#define AFX_ECHAR_H__84E53224_3C47_4247_ABE2_2D44193684BB__INCLUDED_
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
#include "Pchar.h"
#include "Schar.h"
class Echar
{
public:
		Pchar *c3; /*	for C3 plants	*/
		Pchar *c4; /*	for C4 plants	*/
		Schar *soil; /*	for soil organic matter	*/
public:
		Echar();virtual ~Echar();
};
#endif

// Echar.cpp: implementation of the Echar class.
#include "stdafx.h"
#include "SimCycleMFC_OP.h"
#include "Echar.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

Echar::Echar()
{
		c3 = new Pchar();c4 = new Pchar();soil = new Schar();
}
Echar::~Echar()
{		delete c3;	delete c4;	delete soil;}
```

```
// Gchar.h: interface for the Gchar class.
#if !defined(AFX_GCHAR_H__5BD389C1_9A02_4A0F_B4E9_8EEB049E01A8__INCLUDED_)
#define AFX_GCHAR_H__5BD389C1_9A02_4A0F_B4E9_8EEB049E01A8__INCLUDED_
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
class Gchar
{
public:
        int m_iSTFlag; // if 0, stable; if 1, transition

public:
        long        *mm;            /* number of days in each month */
        long        raw, col;       /* grid order, raw and column in 0.5 degree grid */
        long        validation_site;
        long        country;    /* country number */
        long        region;     /* continental region number */
        long        nnn;        /* cell numbers from the origin */
        long        vegNo;      /* number of biomes under processing */
        double      area;       /* biome area for each cell */
        long        year;       /* calculation time from the simulation onset, in year */
        long        month;      /* month of the year, from Jan. to Dec., 0 to 11 */
        long        time_hyd;   /* time to reach stabilization of water budget */
        long        time;       /* time to reach stabilization of carbon budget */
        double      cell_area,land_area;    /* total area and land area for each cell */
        long        CO2y;
        double      *aCO2;      /* atmospheric CO2 concentration, in ppmv */
        double      lat;        /* latitude of the center of grid cell */
        double      lon;        /* longitude of the center of grid cell */
        /* climate condition: *[] means the transitional value */
        double      *tmp_sfc;   /* ground surface temperature, degree Celcius */
        double      tmp_sfc_am;     /* mean temperature, degree Celcius */
        double      tmp_sfc_mx;     /* maximum temperature, degree Celcius */
        double      tmp_sfc_mn;     /* minimum temperature, degree Celcius */
        double      tmp_sfc_bio;
        double      *tmp_2m;        /* 2m air temperature, degree Celcius */
        double      *tmp10_soil; /* soil temperature at 10 cm depth, degree Celcius */
        double      *tmp200_soil;/* soil temperature at 200 cm depth, degree Celcius */
        double      *dswrf_toa;  /* downward shortwave radiation at the atmosphere-top, W m-2 */
        double      *dswrf_sfc;  /* downward shortwave radiation at the surface, W m-2 */
        double      *tcdc_clm;   /* total cloudiness, fraction */
        double      *xprate_sfc; /* precipitation, mm mon-1 */
        double      xprate_sfc_ann; /* precipitation, mm mon-1 */
        double      *spfh_2m;       /* specific humidity, kg kg-1 */
        double      *soilw10;       /* soil moisture content at 10 cm depth, mm */
        double      *soilw200;      /* soil moisture content at 200 cm depth, mm */
        double      *wnd_10m;       /* wind velocity, m s-1 */
        /* climate condition: *_a[] means the average during 1965 to 1998 */
        double      *tmp_sfc_a;     /* ground surface temperature, degree Celcius */
        double      *tmp_2m_a;      /* 2m air temperature, degree Celcius */
        double      *tmp10_soil_a;      /* soil temperature at 10 cm depth, degree Celcius */
        double      *tmp200_soil_a;     /* soil temperature at 200 cm depth, degree Celcius */
        double      *dswrf_toa_a;       /* downward shortwave radiation at the top of atmosphere, W m-2
*/      double      *dswrf_sfc_a;       /* downward shortwave radiation at the surface, W m-2 */
        double      *tcdc_clm_a;        /* total cloudiness, fraction */
        double      *xprate_sfc_a;      /* precipitation, mm mon-1 */
        double      *spfh_2m_a;     /* specific humidity, kg kg-1 */
        double      *soilw10_a;     /* soil moisture content at 10 cm depth, mm */
        double      *soilw200_a;    /* soil moisture content at 200 cm depth, mm */
        double      *wnd_10m_a;     /* wind velocity, m s-1 */
        double      topo;   /* topography, orology, and altitude, m above MSL */
        double      whc;    /* soil water holding capacity */
        double      whc30;  /* soil water holding capacity of above 300mm soil */
```

```
		double	wilt_point;	/*	plant wilting point	*/
		int m_iFutureSim;
public:
		void SetAtmCO2();void IncAtmCO2();Gchar();virtual ~Gchar();
};
#endif
// Gchar.cpp: interface for the Gchar class.
#include "stdafx.h"
#include "SimCycleMFC_OP.h"
#include "Gchar.h"
#include <math.h>
#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif
Gchar::Gchar()
{
		m_iSTFlag= 0 ;		m_iFutureSim=0;
		mm = (long*)calloc(DAYNUM, sizeof(long));	aCO2 = (double*)calloc(DAYNUM, sizeof(double));
		tmp_sfc = (double*)calloc(DAYNUM, sizeof(double));tmp_2m = (double*)calloc(DAYNUM, sizeof(double));
		tmp10_soil = (double*)calloc(DAYNUM, sizeof(double));
		tmp200_soil = (double*)calloc(DAYNUM, sizeof(double));
		dswrf_toa = (double*)calloc(DAYNUM, sizeof(double));
		dswrf_sfc = (double*)calloc(DAYNUM, sizeof(double));
		tcdc_clm = (double*)calloc(DAYNUM, sizeof(double));
		xprate_sfc = (double*)calloc(DAYNUM, sizeof(double));
		spfh_2m = (double*)calloc(DAYNUM, sizeof(double));
		soilw10 = (double*)calloc(DAYNUM, sizeof(double));
		soilw200 = (double*)calloc(DAYNUM, sizeof(double));
		wnd_10m = (double*)calloc(DAYNUM, sizeof(double));
		tmp_sfc_a = (double*)calloc(DAYNUM, sizeof(double));
		tmp_2m_a = (double*)calloc(DAYNUM, sizeof(double));
		tmp10_soil_a = (double*)calloc(DAYNUM, sizeof(double));
		tmp200_soil_a = (double*)calloc(DAYNUM, sizeof(double));
		dswrf_toa_a = (double*)calloc(DAYNUM, sizeof(double));
		dswrf_sfc_a = (double*)calloc(DAYNUM, sizeof(double));
		tcdc_clm_a = (double*)calloc(DAYNUM, sizeof(double));
		xprate_sfc_a = (double*)calloc(DAYNUM, sizeof(double));
		spfh_2m_a = (double*)calloc(DAYNUM, sizeof(double));
		soilw10_a = (double*)calloc(DAYNUM, sizeof(double));
		soilw200_a = (double*)calloc(DAYNUM, sizeof(double));
		wnd_10m_a = (double*)calloc(DAYNUM, sizeof(double));
}
Gchar::~Gchar()
{
		free(mm);free(aCO2);free(tmp_sfc);free(tmp_2m);free(tmp10_soil);
		free(tmp200_soil);free(dswrf_toa);free(dswrf_sfc);free(tcdc_clm);free(xprate_sfc);
		free(spfh_2m);free(soilw10);free(soilw200);free(wnd_10m);free(tmp_sfc_a);
		free(tmp_2m_a);free(tmp10_soil_a);free(tmp200_soil_a);free(dswrf_toa_a);free(dswrf_sfc_a);
		free(tcdc_clm_a);free(xprate_sfc_a);free(spfh_2m_a);free(soilw10_a);free(soilw200_a);
		free(wnd_10m_a);
}
void Gchar::SetAtmCO2()
{
		double base, inc, lgrd, season;
		double amplitude;
		double time,aa0,aa1,aa2,aa3,aa4,aa5;
		time=(double)(CO2y);
		/** BASE **/
		aa0=1904299.0;
		aa1=-3322.4242*pow(time, 1.0);
		aa2=1.6541596*pow(time, 2.0);
		aa3=1.3362655*pow(time, 3.0)/10000.0;
		aa4=-3.0828809*pow(time, 4.0)/10000000.0;
```

```
		aa5=6.2121261*pow(time, 5.0)/100000000000.0;
		base=aa0+aa1+aa2+aa3+aa4+aa5;
		/** INC **/
		inc=0.0;
		/** LGRD **/
		lgrd=1.6*(lat/85.0);
		/** SEASON **/
		amplitude=exp(0.04*lat);
		if(lat>=0.0){
			season=amplitude/2.0*sin(((double)(month)-0.0)/12.0*2.0*PI);
		}else if(lat<0.0){
			season=amplitude/2.0*sin(((double)(month)+6.0)/12.0*2.0*PI);
		}
		aCO2[month]=base+lgrd+season+inc;
}
void Gchar::IncAtmCO2()
{		for(int i=0;i<12; i++){	aCO2[i] += 3.2; }}
```

```
// Pchar.h: interface for the Pchar class.
#if !defined(AFX_PCHAR_H__433B7E55_6BF4_4BFF_8E07_5ADCBDA87D42__INCLUDED_)
#define AFX_PCHAR_H__433B7E55_6BF4_4BFF_8E07_5ADCBDA87D42__INCLUDED_
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
class Schar;class Gchar;class Pmas;class Gcon;class Pflx;
class Pchar
{
public:
        double *m_dDR;double *m_dTest_ci;double *m_dTest_cmp; double *m_dTest_ftem;
        double *m_dTest_fstl;double *m_dTest_fnstl;// phenology
        double m_dLFProp;
public:
        double    albedo;    /* reflectivity, or albedo */
        /*** allocation ***/
        double    *opt_lai;  /* optimum leaf area index */
        double    alloc_ass; /* allocation ration for assimilation organ, fraction */
        double    alloc_abg; /* allocation ration for aboveground non-assimilation organ, fraction */
        double    *malloc_f, *malloc_c, *malloc_r;  /* monthly allocation ratios */
        /*** phenology ***/
        double    gdd, *mgdd;/* growing degree days */
        double    grw_pd;    /* growing period */
        long      *season;/* phenological stage as a function of season */
        /* 0: dormancy */
        /* 1: vegetative growth */
        /* 2: emergence of new leaf */
        /* 3: abandon of old leaf */
        long      frag_emg, frag_dcd;
        /*** photosynthesis ***/
        long      phototype; /** photosynthetic metabolic pathway, 3=C3, 4=C4, 5=CAM **/
        double    pmax;      /* potential maximum rate, micro mol CO2 m-2 s-1 */
        double    ptop;      /* canopy-top photosynthetic rate */
        double    *psat;     /* light-saturated rate, micro mol CO2 m-2 s-1 */
        double    sla;       /* specific leaf area, cm2 g dm-1 */
        double    eK0;       /* light attenuation coefficient, no dimension */
        double    *eK;       /* light attenuation coefficient, no dimension */
        double    lue0;      /* control light dependence coefficient, mol CO2 mol photon-1 */
        double    *lue;      /* monthly quantum yield , mol CO2 mol photon-1 */
        double    topt, topt0; /* optimum temperature, deg C*/
        double    tmin;      /* minimum temperature, deg C*/
        double    tmax;      /* maximum temperature, deg C*/
        double    *ci;       /* monthly intercellular CO2 concentration, ppmv */
        double    kmci;      /* dependence of photosynthesis on intercellular CO2 concentration, ppmv */
        double    cmpcd0; /* CO2 compensation point, ppmv */
        double    *cmpcd; /* CO2 compensation point, ppmv */
        double    *gs;       /* monthly stomatal conductance, mmol H2O m-2 s-1 */
        double    *gc;       /* monthly canopy conductance, mmol H2O m-2 s-1 */
        double    gs_b0,gs_b1,gs_b2; /* parameters of stomatal conductance, mmol H2O m-2 s-1 */
        double    km_nstl;   /* maximum stomatal conductance */
        /*** respiration ***/
        double    rgf, rgc, rgr;  /* specific growth respiration rate, g C g C-1 alloc */
        double    rmf, rmc, rmr;  /* specific maintenance respiration rate at 15 degC, mg C g C-1 day-1 */
        double    rmf0;      /* specific maintenance respiration rate at 15 degC, mg C g C-1 day-1 */
        double    rmc_s, rmr_s; /* specific maintenance respiration rate of sapwood at 15 degC, mg C g C-1 day-1 */
        double    rmc_h, rmr_h; /* specific maintenance respiration rate of heart wood at 15 degC, mg C g C-1 day-1 */
        double    qTf0, qTc0, qTr0; /* temperature dependence, dimensionless */
        double    *qTf, *qTc, *qTr; /* temperature dependence, dimensionless */
                  /*** litter fall ***/
        double    lf0;
        double    lc0;
        double    lr0;       /* specific litter fall rate, fraction */
        double    *lf;
```

```
        double      *lc;
        double      *lr;      /*  specific  litter fall  rate,  fraction  */
        double      dcd;      /*  deciduous  leaf  fraction  */
                    /***  root  stratification  ***/
        double      root_strat;              /*  root  profile  parameter  */
public:
        void pc_sat(Gchar *grid, Gcon *loct);
        void mortality(Gchar *grid);
        void spcfc_res_mass(Gchar *grid,  Pmas *mass);
        void qten_ar(Gchar *grid);
        void F_opt_lai(Gchar *grid,Gcon *loct);
        double canopy_cond( Gchar *grid,Gcon *loct,  Pmas *mass);
        void stom_cond( Gchar *grid,Gcon *loct);
        void quantum_yield(Gchar *grid);
        void incel_cdc(Gchar *grid,Gcon *loct);
        double irr_attn(Gchar  *grid,  Gcon  *loct);
        double lai_mass(Pmas *mass);
        void EcophysiologyVeg(Gchar *grid, Gcon *loct, Pmas *mass);
public:
        void phenology_grass(Gchar *grid, Gcon  *loct);
        void GrowthPeriod(Gchar  *grid,  Gcon  *loct);
        void MON_GrowthPeriod(Gchar  *grid,  Gcon  *loct);
public:
        void leafemergence(  Gchar *grid,     Gcon *loct, Pmas *mass,  Pflx *flux);
        void interval(  Gchar *grid,     Gcon *loct, Pmas *mass,  Pflx *flux);
        void noleafperiod(  Gchar *grid,     Gcon *loct,  Pmas *mass,  Pflx *flux);
        void leaffall(  Gchar *grid,     Gcon *loct,  Pmas *mass,  Pflx *flux);
        void greenperiod( Gchar *grid,     Gcon *loct, Pmas *mass,  Pflx *flux);
        void grass_process( Gchar *grid,  Gcon *loct,  Pflx *flux, Pmas *mass);
        double FromRoot( Gchar *grid,  Gcon *loct,  Pmas *mass);
        double FromStem( Gchar *grid,  Gcon *loct,  Pmas *mass);
        double FromFoliage(Gchar *grid, Gcon *loct,  Pmas *mass);
        double fgpp(Gchar *grid,  Gcon *loct,  Pmas *mass);
public:
        double StemGrowthRes( Gchar *grid,     Gcon *loct,  Pmas *mass,  Pflx *flux);
        double RootGrowthRes( Gchar *grid,     Gcon *loct,  Pmas *mass,  Pflx *flux);
        double RootMaintenRes( Gchar *grid,     Gcon *loct, Pmas *mass);
        double StemMaintenRes( Gchar *grid,     Gcon *loct, Pmas *mass);
        double FoliGrowthRes( Gchar *grid,  Gcon *loct,  Pmas *mass, Pflx *flux);
        double FoliMaintenRes( Gchar *grid,  Gcon *loct, Pmas *mass);
public:
        void DoLitterFall( Gchar *grid,     Gcon *loct, Pmas *mass,  Pflx *flux);
        void SetParameterC3(Gchar *grid);
        void SetParameterC4(Gchar *grid);
        Pchar();
        virtual ~Pchar();
};
#endif

// Pchar.cpp: interface for the Pchar class.
#include "stdafx.h"
#include "SimCycleMFC_OP.h"
#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif
Pchar::Pchar()
{
    m_dDR = (double*)calloc(DAYNUM, sizeof(double));
        //test variable
        m_dTest_ci = (double*)calloc(DAYNUM, sizeof(double));
        m_dTest_cmp = (double*)calloc(DAYNUM, sizeof(double));
        m_dTest_ftem = (double*)calloc(DAYNUM, sizeof(double));
        m_dTest_fstl = (double*)calloc(DAYNUM, sizeof(double));
```

```
		m_dTest_fnstl = (double*)calloc(DAYNUM, sizeof(double));
			m_dLFProp = 0.3;
		opt_lai = (double*)calloc(DAYNUM, sizeof(double));
		malloc_f = (double*)calloc(DAYNUM, sizeof(double));
		malloc_c = (double*)calloc(DAYNUM, sizeof(double));
		malloc_r = (double*)calloc(DAYNUM, sizeof(double));
		mgdd = (double*)calloc(DAYNUM, sizeof(double));
		psat = (double*)calloc(DAYNUM, sizeof(double));
		eK = (double*)calloc(DAYNUM, sizeof(double));
		lue = (double*)calloc(DAYNUM, sizeof(double));
		ci = (double*)calloc(DAYNUM, sizeof(double));
		cmpcd = (double*)calloc(DAYNUM, sizeof(double));
		gs = (double*)calloc(DAYNUM, sizeof(double));
		gc = (double*)calloc(DAYNUM, sizeof(double));
		qTf = (double*)calloc(DAYNUM, sizeof(double));
		qTc = (double*)calloc(DAYNUM, sizeof(double));
		qTr = (double*)calloc(DAYNUM, sizeof(double));
		lf = (double*)calloc(DAYNUM, sizeof(double));
		lc = (double*)calloc(DAYNUM, sizeof(double));
		lr = (double*)calloc(DAYNUM, sizeof(double));
		season = (long*)calloc(DAYNUM, sizeof(long));
}
Pchar::~Pchar()
{
		free(opt_lai);free(malloc_f);free(malloc_c);free(malloc_r);free(mgdd);free(psat);
		free(eK);free(lue);free(ci);free(cmpcd);
		free(gs);free(gc);free(qTf);free(qTc);free(qTr);
		free(lf);free(lc);free(lr);	free(season);
}
void Pchar::SetParameterC3(Gchar *grid)
{
		phototype=3;

		albedo= 0.16;alloc_ass= 0.24; alloc_abg= 0.01; sla= 125.0;eK0= 0.38;
		lue0=	0.05; pmax=	24.7;topt0= 22.0;	tmin=	0.0; tmax=	45.0;
		gs_b0=10.0;	gs_b1=17000.0; gs_b2= 5.00; km_nstl= 0.6; kmci=		40.0;
		cmpcd0= 50.0; rgf= 0.48; rgc= 0.150; rgr=0.290;rmf0=1.55;
		rmc_s=0.100; rmr_s=0.13;rmc_h =0.0880; rmr_h=0.3210;qTf0= 2.0;
		qTc0=2.0;qTr0= 2.0;lf0=3.460+ 3.5; lc0=0.2300 * 2;lr0= 1.400/5.5;
		dcd= 0.70;root_strat=0.95;
}
void Pchar::SetParameterC4(Gchar *grid)
{
		phototype=4;
		albedo= 0.16;alloc_ass= 0.24; alloc_abg= 0.01; sla= 125.0;eK0= 0.38;
		lue0=	0.05; pmax=	24.7;topt0= 22.0;	tmin=	0.0; tmax=	45.0;
		gs_b0=10.0;	gs_b1=17000.0; gs_b2= 5.00; km_nstl= 0.6; kmci=		40.0;
		cmpcd0= 50.0; rgf= 0.48; rgc= 0.150; rgr=0.290;rmf0=1.55;
		rmc_s=0.100; rmr_s=0.13;rmc_h =0.0880; rmr_h=0.3210;qTf0= 2.0;
		qTc0=2.0;qTr0= 2.0;lf0=3.460+ 3.5; lc0=0.2300 * 2;lr0= 1.400/5.5;
		dcd= 0.70;root_strat=0.95;
}
void Pchar::pc_sat(Gchar *grid, Gcon *loct)
{
		double	ftem,fstl,fnstl;double	crit_sw;double	aa1,aa2,aa3,bb1;
		ftem=fstl=fnstl=1.0;
		/**	optimum	photosynthesis	temperature	**/
		if(phototype==3){	/*	C3	plants	with	change	*/
			topt= topt0 + 0.01 * ci[grid->month];
		}else if(phototype==4){	/*	C4	plants	without	change	*/
			topt=topt0;
		}
		/**	CO2	compensation	point	**/
		if(phototype==3){	/*	C3	plants	with	change:	Brooks&Farquhar(1985)	*/
			aa3 = 0.000347*(grid->tmp_sfc[grid->month]-20.0)*(grid->tmp_sfc[grid->month]-20.0);
```

```
		aa1 = 1.0+0.0451*(grid->tmp_sfc[grid->month]-20.0)+aa3;
		aa1 = (aa1>0.0)?aa1:0.0;
		cmpcd[grid->month]=cmpcd0*aa1;
	}else if(phototype==4){	/* C4 plants without change */
		cmpcd[grid->month]=cmpcd0;
	}
	/** temperature effect on photosynthesis **/
	aa1 = (grid->tmp_sfc[grid->month]-tmax)*(grid->tmp_sfc[grid->month]-tmin);
	aa2 = (grid->tmp_sfc[grid->month]-topt)*(grid->tmp_sfc[grid->month]-topt);
	ftem = aa1/(aa1-aa2);
	ftem = (ftem<=1.0)?ftem:1.0;	ftem=(ftem>=0.0)?ftem:0.0;
	/** stomatal limitation on photosynthesis via intercellular CO2 concentration **/
	fstl=0.2+0.8*(ci[grid->month]-cmpcd[grid->month])/(kmci+ci[grid->month]);
	if(phototype == 4) fstl=1;

	m_dTest_ci[grid->month]=ci[grid->month];
	m_dTest_cmp[grid->month]=cmpcd[grid->month];
	m_dTest_fstl[grid->month]=fstl;
	fstl=(fstl<=1.0)?fstl:1.0;
	fstl=(fstl>=0.0)?fstl:0.0;
	/** non-stomatal limitation on photosynthesis **/
	if(grid->vegNo == 2 ){
		double dCurveSlop = 0.04;
		double dCompan = -0.25;
		fnstl = dCurveSlop * loct->sww / (dCurveSlop * loct->sww + km_nstl) - dCompan;
	}else{
		if(grid->whc>0.0){
			crit_sw = grid->wilt_point/grid->whc;
		}else{
			crit_sw=0.10;
		}
		crit_sw=(crit_sw<=0.5)?crit_sw:0.5;	crit_sw=(crit_sw>=0.05)?crit_sw:0.05;
		bb1= loct->sww + grid->whc * km_nstl;
		if(phototype==3){	/* C3 plants */
			if(bb1>0.0){
				fnstl=(1.0 - crit_sw) * loct->sww/bb1 + crit_sw;	/**/
			}else{
				fnstl=0.0;
			}
		}else if(phototype==4){	/* C4 plants */
			if(bb1>0.0){
				fnstl=(1.0 - crit_sw) * loct->sww/bb1 + crit_sw;	/**/
			}else{
				fnstl=0.0;
			}
		}
		fnstl=(fnstl<=1.0)?fnstl:1.0;	fnstl=(fnstl>=0.0)?fnstl:0.0;
	}
	/** light-saturated photosynthesis rate **/
	psat[grid->month]=pmax * ftem * fstl * fnstl;	//pmax*ftem*fstl*fnstl;
	m_dTest_ftem[grid->month]=ftem;
	m_dTest_fnstl[grid->month]=fnstl;
}
/******* ecophysiological vegetation processes *******/
void Pchar::EcophysiologyVeg( Gchar *grid, Gcon *loct, Pmas *mass)
{
	long g;
	double aaa,bbb;
	/* give leaf area index (LAI), m2 m-2*/
	mass->lai[grid->month]=lai_mass(mass);
	/* give irradiance attenuation coefficient */
	eK[grid->month]=irr_attn(grid, loct);
	ci[grid->month]=grid->aCO2[grid->month]*0.5;	/** initial ci **/
	/* stabilization of single-leaf processes */
	for(g=0;g<6;g++){
```

```
                /* give quantum yield , mol CO2 mon photon-1*/
                quantum_yield(grid);
                /* give light-saturated photosynthesis rate, micromol m-2 s-1 */
                pc_sat(grid, loct);
                /* canopy-top photosynthetic rate*/
                aaa=psat[grid->month] * lue[grid->month]*loct->par[grid->month];
                bbb=psat[grid->month] + lue[grid->month]*loct->par[grid->month];
                if(bbb>0.0){
                        ptop=aaa/bbb;
                }else{
                        ptop=0.0;
                }
                /* give stomatal conductance, mmol H2O m-2 s-1 */
                stom_cond(grid, loct);
                /* give intercellular CO2 concentration, ppmv */
                incel_cdc(grid, loct);
        }
        /** canopy conductance **/
        gc[grid->month] = canopy_cond(grid, loct, mass);
        /** plant respiration **/
        qten_ar(grid); /*Q10*/
        spcfc_res_mass(grid, mass); /* woody specific respiration rate */
        /** litterfall of plant respiration **/
        mortality(grid);
        /** optimum leaf area index **/
        F_opt_lai(grid,loct);
}
double Pchar::lai_mass( Pmas *mass)
{
        double sla,lai_est;
        /** specific leaf area as a function of... what? **/
        sla=this->sla;
        lai_est= sla * mass->fol * dmTc/100.0/2.0; //#define dmTc 2.2
        lai_est=(lai_est>=0.0)?lai_est:0.0;
        /* dmTc: dry-matter to carbon */
        /* 100.0: cm2 g dm-1 to Mg ha-1 base */
        /* 2.0: single-sided leaf area */
        return(lai_est);
}
/********** give irradiance attenuation coefficient *************/
double Pchar::irr_attn( Gchar *grid, Gcon *loct)
{
        double aaa,bbb;
        /* a function of solar hight angle */
        aaa=sin(loct->sl_hgt[grid->month]*dTr);
        aaa=(aaa<=1.0)?aaa:1.0;  aaa=(aaa>=0.3)?aaa:0.3; /* to avoid extreme values*/
        bbb=eK0/aaa; //eK0 : 0.4
        return(bbb);
}
/********** intercellular CO2 concentration *************/
void Pchar::incel_cdc( Gchar *grid, Gcon *loct)
{
        double gs_co2;
        /* give intercellular CO2 concentration, as a function of ambient CO2 level and stomatal
conductance */
        gs_co2 = gs[grid->month]/1.56;
        /* 1.56: conversion from H2O to CO2 conductance */
        double dci = grid->aCO2[grid->month]-(ptop/(gs_co2/1000.0));
        /* 1000.0: conbert from mmol to 탑ol */
        dci = (dci >= 0.0)?dci:0.0;
        dci = (dci <= grid->aCO2[grid->month]) ? dci : grid->aCO2[grid->month];
        ci[grid->month] = dci;
}
/*********** quantum yield C3 and C4 *************/
void Pchar::quantum_yield( Gchar *grid)
```

```
{
        double eftem, efci;
        /** plant_type: 3=C3,       4=C4,     (5=CAM)       **/
        if(phototype==3){
                /* temperature dependence */
                eftem=(52.0-grid->tmp_sfc[grid->month])/(3.5+0.75*(52.0-grid->tmp_sfc[grid->month]));
                /* CO2 dependence */
                efci= 0.75* ci[grid->month]/(10.0+0.6*ci[grid->month]);
                /* 3.5, 52.0, etc.: empirical parameters */
        }else if(phototype==4){
                /* insensitive QE of C4 species */
                eftem=1.0;
                efci=1.0;
        }
        /* give quantum yield */
        lue[grid->month]=lue0*eftem*efci;
}
/************ stomatal conductance **************/
void Pchar::stom_cond(Gchar *grid, Gcon *loct)
{
        double b1d,cc;
        /* stomatal conductance model by Ball, Woodraw, and Berry (1987) */
        b1d = gs_b1 /
                ((grid->aCO2[grid->month] - cmpcd[grid->month]) * (1.0 + loct->vpd[grid->month]/gs_b2));

        /** add soil water factor **/
        cc=1.0;  /* not defined yet */
        if(psat[grid->month]>0.0){
                gs[grid->month]=gs_b0 + b1d * ptop ;
        }else{
                gs[grid->month]=0;
        }
}
/********** canopy conductance **************/
double Pchar::canopy_cond( Gchar *grid, Gcon *loct,     Pmas *mass)
{
        double aaa,sss,ttt,uuu,vvv,lue_gs, canopy_cond;
        /* NOTE: integrate leaf stomatal conductance with considering light attenuation in the canopy */
        aaa = gs_b0 + gs_b1 / (grid->aCO2[grid->month] - cmpcd[grid->month]);
        lue_gs = lue[grid->month] * (aaa/pmax);

        if(mass->lai[grid->month]>0.0){
                sss= 2.0 * gs[grid->month] / eK[grid->month];
                ttt= 1.0 + sqrt(1.0+eK[grid->month]*lue_gs*loct->par[grid->month]/gs[grid->month]);
                vvv= -1.0 * eK[grid->month]*mass->lai[grid->month];
                uuu= 1.0 + sqrt(1.0+eK[grid->month]*lue_gs*loct->par[grid->month]*exp(vvv)/gs[grid->month]);
                canopy_cond = sss * log(ttt/uuu);
        }else{
                /* no leaf, no conductance*/
                canopy_cond=0.0;
        }
        return gs[grid->month]*mass->lai[grid->month];
}
/************* optimum LAI by Kuroiwa (1966) **************/
void Pchar::F_opt_lai( Gchar *grid, Gcon *loct)
{
        double aaa,bbb,ccc,cc4,ddd, eee;
        double arm, arg, ar;
        aaa= 1.0/eK[grid->month];
        bbb= eK[grid->month] * lue[grid->month] * loct->par[grid->month];
        /* daily respiratory cost */
        eee= log(qTc[grid->month]) / 10.0 * (grid->tmp_sfc[grid->month] - 15.0);
        arm= rmf * exp(eee)/1000.0 * dmTc * 10000.0/(sla);
        arg= lf[grid->month]/1000.0 * dmTc * 10000.0/(sla)*(1.0+rgf);
        ar= arm + arg;
```

```
		if(psat[grid->month]>0.0){
			cc4=(psat[grid->month]*loct->dlen[grid->month])/(psat[grid->month]*loct->dlen[grid->month]-ar*24.0);
			ccc=psat[grid->month]*(cc4-1.0);

			ddd=bbb/ccc;
			ddd=(ddd>1.0)?ddd:1.0;

			opt_lai[grid->month]=aaa*log(ddd);
		}else{
			opt_lai[grid->month]=0.0;
		}
}

/***********	Q10	of	autotrophic	respiration	****************/
void Pchar::qten_ar( Gchar *grid)
{
		double aaa;
		/*	larger	at	cool	and	smaller	at	warm	*/
		aaa=exp(-0.009*(grid->tmp_sfc[grid->month]-15.0));
		qTf[grid->month]=qTf0*aaa;qTc[grid->month]=qTc0*aaa;qTr[grid->month]=qTr0*aaa;
}
void Pchar::spcfc_res_mass(Gchar *grid, Pmas *mass)
{
		double powstm,powrot;  double stm_sap,stm_hrt,rot_sap,rot_hrt;
		rmf=rmf0;
		/*	specific	respiration	increasing	in	a	power	of	2/3	manner */
		powstm = 1.0 - 0.33334 * mass->stm / (50.0+mass->stm);
		powrot = 1.0 - 0.33334 * mass->rot / (50.0+mass->rot);
		stm_sap = pow(mass->stm, powstm);	/*	sapwood	mass	in	stem	*/
		stm_hrt = mass->stm - stm_sap;		/*	heartwood	mass	in	stem	*/
		rot_sap = pow(mass->rot, powrot);	/*	sapwood	mass	in	root	*/
		rot_hrt = mass->rot - rot_sap;		/*	heartwood	mass	in	root	*/
		rmc = (rmc_s * stm_sap + rmc_h * stm_hrt)/(mass->stm+0.00001);
		rmr = (rmr_s*rot_sap+rmr_h*rot_hrt)/(mass->rot+0.00001);
}
/************	mortarity	of	plant	organs	**************/
void Pchar::mortality( Gchar *grid)
{
		double bbb=1.0;
		/*	larger	at	warm,	and	smaller	at	cool	*/
		/*	bbb=1.0+(grid->tmp_sfc[grid->month]-grid->tmp_sfc_am)/100.0;	*/
		lf[grid->month] = lf0 * bbb; /* leaf */
		lc[grid->month] = lc0 * bbb; /* stem */
		lr[grid->month] = lr0 * bbb; /* root */
}
void Pchar::GrowthPeriod(Gchar *grid, Gcon	*loct)
{
		/**	cumulative	temperature,	degree	days**/
		if(grid->lat>=0.0){
			if(grid->month==0){
				gdd=0.0;
				if(grid->year==0){
					frag_dcd=1;frag_emg=0;
				}
			}
		}else if(grid->lat<0.0){
			if(grid->month==6){
				gdd=0.0;
				if(grid->year==0){
					frag_dcd=0;frag_emg=1;
				}
			}
		}
		if(grid->tmp_sfc[grid->month]>5.0){
			gdd += grid->tmp_sfc[grid->month] * (double)grid->mm[grid->month];
```

```
		}
		mgdd[grid->month]=gdd;
		/**	growing	period,	days	**/
		if(grid->month==0) grw_pd=0.0;
		if(grid->tmp_sfc[grid->month]>5.0){
			grw_pd+=(double)grid->mm[grid->month];
		}
		/**	phenology	**/
		phenology_grass(grid, loct);
}
void Pchar::MON_GrowthPeriod(Gchar *grid, Gcon	*loct)
{
		/**	cumulative	temperature,	degree	days**/
		if(grid->lat>=0.0){
			if(grid->month==0){
				gdd=0.0;
				if(grid->year==0){
					frag_dcd=1;frag_emg=0;
				}
			}
		}else if(grid->lat<0.0){
			if(grid->month==6){
				gdd=0.0;
				if(grid->year==0){
					frag_dcd=0;frag_emg=1;
				}
			}
		}
		if(grid->tmp_sfc[grid->month]>5.0){
			gdd += grid->tmp_sfc[grid->month] * (double)grid->mm[grid->month];
		}
		mgdd[grid->month]=gdd;
		/**	growing	period,	days	**/
		if(grid->month==0) grw_pd=0.0;
		if(grid->tmp_sfc[grid->month]>5.0){
			grw_pd+=(double)grid->mm[grid->month];
		}
		/**	phenology	**/
		phenology_grass(grid, loct);
}
/*******	grassland	*******/
void Pchar::phenology_grass(Gchar *grid, Gcon	*loct)
{
		if(phototype==3){
			double iWaterStress =0;
			double iC3Temp=0;
			if(grid->vegNo == 2){
				iWaterStress = 0.002;iC3Temp = 5.;
			}else {
				iWaterStress = 0.05;iC3Temp = 5.0;
			}
			if(loct->msww[grid->month]/grid->whc < iWaterStress ||
				grid->tmp_sfc[grid->month] < iC3Temp){
				/*	dormancy	*/
				season[grid->month]=0;
				/*	leaf-shedding	*/
				if(frag_emg==1 && frag_dcd==0){
					season[grid->month]=3;
					frag_dcd=1;
					frag_emg=0;
				}
			}else if(loct->msww[grid->month]/grid->whc >= iWaterStress &&
					grid->tmp_sfc[grid->month] >= iC3Temp){
				/*	growing-period	*/
				season[grid->month]=1;
```

```
                        /*    leaf-emergence    */
                        if(frag_emg==0){
                                season[grid->month]=2;
                                frag_emg=1;
                                frag_dcd=0;
                        }
                        if(grid->vegNo == 2){
                if(grid->month > 6 && grid->tmp_sfc[grid->month] > 5 && grid->tmp_sfc[grid->month + 1] < 5){
                                        season[grid->month] = 4;m_dLFProp= 0.7;
                                }
                        }
                }
        }else if(phototype==4){
                if(loct->msww[grid->month]/grid->whc<0.1||grid->tmp_sfc[grid->month]<8.0){
                        /*    dormancy    */
                        season[grid->month]=0;
                        /*    leaf-shedding    */
                        if(frag_emg==1&&frag_dcd==0){
                                season[grid->month]=3; frag_dcd=1; frag_emg=0;
                        }
                }else if(loct->msww[grid->month]/grid->whc>=0.1&&grid->tmp_sfc[grid->month]>=8.0){
                        /*    growing-period    */
                        season[grid->month]=1;
                        /*    leaf-emergence    */
                        if(frag_emg==0){
                                season[grid->month]=2; frag_emg=1;frag_dcd=0;
                        }
                }
        }
}

void Pchar::grass_process( Gchar *grid,    Gcon *loct,    Pflx *flux, Pmas *mass)
{
        flux->beforedeal(grid,loct,this,mass);
        switch(season[grid->month]){
                case 0: /*    dormancy    */
                        noleafperiod(grid,loct,mass,flux);      break;
                case 1:    /*    vegetative    growth    */
                        greenperiod(grid,loct,mass,flux);
                        if(grid->vegNo != 2) flux->reallocation_survival(grid, this, mass);break;
                case 2:    /*    leaf emergence    */
                        leafemergence(grid,loct,mass,flux);    break;
                case 3:    /*    leaf shedding    */
                        leaffall(grid,loct,mass,flux);    break;
                case 4:    // leaf shedding start
                        leaffall(grid,loct,mass,flux);    break;
        }
        flux->afterdeal(grid,loct,this,mass);
}
/*************    green    period,    while    plants    grow    up    actively *************/
void Pchar::greenperiod( Gchar *grid,        Gcon *loct, Pmas *mass,    Pflx *flux)
{
        double    nn, area_frac;
        if(phototype==3){
                area_frac=loct->C3ptn;
        }else if(phototype==4){
                area_frac=loct->C4ptn;
                if(area_frac == 0) mass->fol = 0;
        }
        nn=(double)grid->mm[grid->month];
        if(grid->vegNo == 2 ){// if emergency right before
                double bbb=(opt_lai[grid->month]>1.0)?opt_lai[grid->month]:1.0;
                double emerge=(bbb-mass->lai[grid->month])*100.0*2.0/2.2/sla;
                double aaa= mass->rot;
                // transition of root biomass to shoot
```

```
                if(season[grid->month - 1] == 2) emerge=(emerge>aaa*0.007)?aaa*0.007:emerge;
                else if(season[grid->month - 2] == 2) emerge=(emerge>aaa*0.03)?aaa*0.03:emerge;
                else if(season[grid->month - 3] == 2)emerge=(emerge>aaa*0.025)?aaa*0.025:emerge;
                else emerge=(emerge>aaa*0.001)?aaa*0.001:emerge;
                if(aaa>0.0){
                        double dProp= 0.8;
                        mass->fol += dProp * emerge;
                        mass->stm += (1 - dProp ) * emerge;
                        mass->rot -= emerge;
                }
        }
        mass->lai[grid->month]= lai_mass(mass);
        /*  photosynthesis,  gross  primary  production  */
        flux->gpp[grid->month]=nn* fgpp(grid, loct, mass);
        /*  maintenance respirations  */
        flux->rfm[grid->month]=nn* FoliMaintenRes(grid, loct, mass);
        flux->rcm[grid->month]=nn* StemMaintenRes(grid, loct, mass);
        flux->rrm[grid->month]=nn* RootMaintenRes(grid, loct, mass);
        flux->rpm[grid->month]=flux->rfm[grid->month]+flux->rcm[grid->month]+flux->rrm[grid->month];
        /*  tentative  primary  production  */
        flux->epp[grid->month]=flux->gpp[grid->month]-flux->rpm[grid->month];
        /*  translocation  of  photosynthate  */
        flux->allocation(grid, this,  mass);
        if(flux->epp[grid->month]>0.0){
                /*  growth construction  respiration  */
                flux->rfg[grid->month]= FoliGrowthRes(grid, loct, mass, flux);
                flux->rcg[grid->month]= StemGrowthRes(grid, loct, mass, flux);
                flux->rrg[grid->month]= RootGrowthRes(grid, loct, mass, flux);
        }else if(flux->epp[grid->month]<=0.0){
                flux->rfg[grid->month]=flux->rcg[grid->month]=flux->rrg[grid->month]=0.0;
        }
        /*  partitioning  of  photosynthate */
        mass->fol += (flux->tpf[grid->month]-flux->rfg[grid->month])*area_frac;
        mass->rot += (flux->tpr[grid->month]-flux->rrg[grid->month])*area_frac;
        mass->stm += (flux->tpc[grid->month]-flux->rcg[grid->month])*area_frac;
        //Litter fall
        DoLitterFall(grid, loct, mass, flux);
}

/*************  leaf shedding  *************/
void Pchar::leaffall( Gchar *grid,      Gcon *loct, Pmas *mass,   Pflx *flux)
{
        double  nn, area_frac;
        if(phototype==3){
                area_frac=loct->C3ptn;
        }else if(phototype==4){
                area_frac=loct->C4ptn;
        }
        nn=(double)grid->mm[grid->month];
        //Litter fall
        DoLitterFall(grid, loct, mass, flux);
        mass->lai[grid->month]= lai_mass(mass);
        /*  photosynthesis,  gross  primary  production  */
        flux->gpp[grid->month]=nn* fgpp(grid, loct, mass);
        /*  maintenance respirations  */
        flux->rfm[grid->month]=nn* FoliMaintenRes(grid, loct, mass);
        flux->rcm[grid->month]=nn* StemMaintenRes(grid, loct, mass);
        flux->rrm[grid->month]=nn* RootMaintenRes(grid, loct, mass);
        flux->rpm[grid->month]=flux->rfm[grid->month]+flux->rcm[grid->month]+flux->rrm[grid->month];
        /*  tentative  primary  production  */
        flux->epp[grid->month]=flux->gpp[grid->month]-flux->rpm[grid->month];
        /*  translocation  of  photosynthate  */
        flux->allocation(grid, this,  mass);
        if(flux->epp[grid->month]>0.0){
                /*  growth construction  respiration  */
```

```
		flux->rfg[grid->month]= FoliGrowthRes(grid, loct, mass, flux);
		flux->rcg[grid->month]= StemGrowthRes(grid, loct, mass, flux);
		flux->rrg[grid->month]= RootGrowthRes(grid, loct, mass, flux);
	}else if(flux->epp[grid->month]<=0.0){
		flux->rfg[grid->month]=flux->rcg[grid->month]=flux->rrg[grid->month]=0.0;
	}
	/*	partitioning	of	photosynthate */
	mass->fol+=(flux->tpf[grid->month]-flux->rfg[grid->month])*area_frac;
	mass->rot+=(flux->tpr[grid->month]-flux->rrg[grid->month])*area_frac;
	mass->stm+=(flux->tpc[grid->month]-flux->rcg[grid->month])*area_frac;
}
/*************	new	leaf	emergence	*************/
void Pchar::leafemergence( Gchar *grid,	Gcon *loct, Pmas *mass,	Pflx *flux)
{
	double	nn,	aaa, bbb, emerge;
	double	area_frac;
	if(phototype==3){
		area_frac=loct->C3ptn;
	}else if(phototype==4){
		area_frac=loct->C4ptn;
	}
	nn=(double)grid->mm[grid->month];
	/*	leaf	emergence	*/
	if(grid->vegNo == 2){
		bbb=(opt_lai[grid->month]>1.0)?opt_lai[grid->month]:1.0;
		emerge=(bbb-mass->lai[grid->month])*100.0*2.0/2.2/sla;
		aaa= mass->rot;
		emerge=(emerge>aaa*0.005)?aaa*0.005:emerge;
		if(aaa>0.0){
			double dProp= 0.9;
			mass->fol += dProp * emerge;
			mass->stm += (1 - dProp ) * emerge;
			mass->rot -= emerge;
		}
	}else {
		bbb=(opt_lai[grid->month]>2.0)?opt_lai[grid->month]:2.0;
		emerge=(bbb-mass->lai[grid->month])*100.0*2.0/2.2/sla;
		aaa=mass->stm+mass->rot;
		emerge=(emerge>aaa*0.3)?aaa*0.3:emerge;
		if(aaa>0.0){
			mass->fol+=emerge;
			mass->stm-=emerge*mass->stm/aaa;
			mass->rot-=emerge*mass->rot/aaa;
		}
	}
	//Litter fall
	DoLitterFall(grid, loct, mass, flux);
	mass->lai[grid->month]= lai_mass(mass);
	/*	photosynthesis,	gross	primary	production	*/
	flux->gpp[grid->month]=nn* fgpp(grid, loct, mass);
	/*	maintenance respirations	*/
	flux->rfm[grid->month]=nn* FoliMaintenRes(grid, loct, mass);
	flux->rcm[grid->month]=nn* StemMaintenRes(grid, loct, mass);
	flux->rrm[grid->month]=nn* RootMaintenRes(grid, loct, mass);
	flux->rpm[grid->month]=flux->rfm[grid->month]+flux->rcm[grid->month]+flux->rrm[grid->month];
	/*	tentative	primary	production	*/
	flux->epp[grid->month]=flux->gpp[grid->month]-flux->rpm[grid->month];
	/*	translocation	of	photosynthate	*/
	flux->allocation(grid, this,	mass);
	if(flux->epp[grid->month]>0.0){
		/*	growth construction	respiration	*/
		flux->rfg[grid->month]= FoliGrowthRes(grid, loct, mass, flux);
		flux->rcg[grid->month]= StemGrowthRes(grid, loct, mass, flux);
		flux->rrg[grid->month]= RootGrowthRes(grid, loct, mass, flux);
	}else if(flux->epp[grid->month]<=0.0){
```

```
                        flux->rfg[grid->month]=flux->rcg[grid->month]=flux->rrg[grid->month]=0.0;
            }
            /*    partitioning    of    photosynthate */
            mass->fol+=(flux->tpf[grid->month]-flux->rfg[grid->month])*area_frac;
            mass->stm+=(flux->tpc[grid->month]-flux->rcg[grid->month])*area_frac;
            mass->rot+=(flux->tpr[grid->month]-flux->rrg[grid->month])*area_frac;
}

/*************    dormancy    period    *************/
void Pchar::noleafperiod( Gchar *grid,        Gcon *loct, Pmas *mass,    Pflx *flux)
{
            double    nn, area_frac;
            if(phototype==3){
                        area_frac=loct->C3ptn;
            }else if(phototype==4){
                        area_frac=loct->C4ptn;
                        if(area_frac == 0) mass->fol = 0;
            }
            nn=(double)grid->mm[grid->month];
            //Litter fall
            DoLitterFall(grid, loct, mass, flux);
            mass->lai[grid->month] = lai_mass(mass);
            /*    photosynthesis,    gross    primary    production    */
            flux->gpp[grid->month]=nn* fgpp(grid, loct, mass);
            /*    maintenance respirations    */
            flux->rfm[grid->month]=nn* FoliMaintenRes(grid, loct, mass);
            flux->rcm[grid->month]=nn* StemMaintenRes(grid, loct, mass);
            flux->rrm[grid->month]=nn* RootMaintenRes(grid, loct, mass);
            flux->rpm[grid->month]=flux->rfm[grid->month]+flux->rcm[grid->month]+flux->rrm[grid->month];
            /*    tentative    primary    production    */
            flux->epp[grid->month]=flux->gpp[grid->month]-flux->rpm[grid->month];
            /*    translocation    of    photosynthate      */
            flux->allocation(grid, this,    mass);
            if(flux->epp[grid->month]>0.0){
                        /*    growth construction    respiration    */
                        flux->rfg[grid->month]= FoliGrowthRes(grid, loct, mass, flux);
                        flux->rcg[grid->month]= StemGrowthRes(grid, loct, mass, flux);
                        flux->rrg[grid->month]= RootGrowthRes(grid, loct, mass, flux);
            }else if(flux->epp[grid->month]<=0.0){
                        flux->rfg[grid->month]=flux->rcg[grid->month]=flux->rrg[grid->month]=0.0;
            }
            /*     partitioning    of    photosynthate */
            mass->fol+=(flux->tpf[grid->month]-flux->rfg[grid->month])*area_frac;
            mass->stm+=(flux->tpc[grid->month]-flux->rcg[grid->month])*area_frac;
            mass->rot+=(flux->tpr[grid->month]-flux->rrg[grid->month])*area_frac;
}

/***     interval    period,    fallowing    lands      ***/
void Pchar::interval(    Gchar *grid,        Gcon *loct, Pmas *mass,    Pflx *flux)
{
            double    nn;
            double    area_frac;
            if(phototype==3){
                        area_frac=loct->C3ptn;
            }else if(phototype==4){
                        area_frac=loct->C4ptn;
            }
            nn=(double)grid->mm[grid->month];
            /*    litter    */
            flux->lf[grid->month]=nn* FromFoliage(grid, loct, mass);
            flux->lc[grid->month]=nn* FromStem(grid, loct, mass);
            flux->lr[grid->month]=nn* FromRoot(grid, loct, mass);
            /*    litter    fall      */
            mass->fol-=flux->lf[grid->month]*area_frac;
            mass->stm-=flux->lc[grid->month]*area_frac;
```

```
		mass->rot-=flux->lr[grid->month]*area_frac;
		/* photosynthesis, gross primary production */
		flux->gpp[grid->month]=nn* fgpp(grid, loct, mass);
		/* maintenance respirations */
		flux->rfm[grid->month]=nn* FoliMaintenRes(grid, loct, mass);
		flux->rcm[grid->month]=nn* StemMaintenRes(grid, loct, mass);
		flux->rrm[grid->month]=nn* RootMaintenRes(grid, loct, mass);
		flux->rpm[grid->month]=flux->rfm[grid->month]+flux->rcm[grid->month]+flux->rrm[grid->month];
		/* tentative primary production */
		flux->epp[grid->month]=flux->gpp[grid->month]-flux->rpm[grid->month];
		/* translocation of photosynthate */
		flux->allocation(grid, this, mass);
		if(flux->epp[grid->month]>0.0){
			/* growth construction respiration */
			flux->rfg[grid->month]= FoliGrowthRes(grid, loct, mass, flux);
			flux->rcg[grid->month]= StemGrowthRes(grid, loct, mass, flux);
			flux->rrg[grid->month]= RootGrowthRes(grid, loct, mass, flux);
		}else if(flux->epp[grid->month]<=0.0){
			flux->rfg[grid->month]=flux->rcg[grid->month]=flux->rrg[grid->month]=0.0;
		}
		/* partitioning of photosynthate */
		mass->fol+=(flux->tpf[grid->month]-flux->rfg[grid->month])*area_frac;
		mass->stm+=(flux->tpc[grid->month]-flux->rcg[grid->month])*area_frac;
		mass->rot+=(flux->tpr[grid->month]-flux->rrg[grid->month])*area_frac;
}
double Pchar::FromFoliage(Gchar *grid, Gcon *loct, Pmas *mass)
{
		/* constant fraction of senescence organ, abandoned as litter */
		if(grid->vegNo == 2 ){
			return lf[grid->month] * mass->fol /1000.0;
		}else if( grid->vegNo == 51 && phototype == 3){//semi desert shrub
			return lf[grid->month] * mass->fol /700.0;
		}else if( grid->vegNo == 4 && phototype == 3){//semi desert shrub
			return lf[grid->month] * mass->fol /330.0;
		}else
			return lf[grid->month] * mass->fol /1100.0;
}
double Pchar::FromStem( Gchar *grid, Gcon *loct, Pmas *mass)
{
		/* constant fraction of senescence organ, abandoned as litter */
		if(grid->vegNo == 2){
			return lc[grid->month]*mass->stm/1000.0;
		}else return lc[grid->month]*mass->stm/1000.0;
}
double Pchar::FromRoot( Gchar *grid, Gcon *loct, Pmas *mass)
{
		/* constant fraction of senescence organ, abandoned as litter */
		if(grid->vegNo == 2){
			return lr[grid->month] * mass->rot/1000.0;
		}else if( grid->vegNo == 4 && phototype == 3){
			return lr[grid->month] * mass->rot/8000.0;
		}else if( grid->vegNo == 51 && phototype == 3){

			return lr[grid->month] * mass->rot/1000.0;
		}else
			return lr[grid->month] * mass->rot/1000.0;
}
double Pchar::fgpp(Gchar *grid, Gcon *loct, Pmas *mass)
{
		double gpp;double cc1,cc2,cc3,bb;
		/* give daily GPP based on the Monsi-Saeki theory, or Kuroiwa's equation */
		if(psat[grid->month]>0.0){
			cc1=2.0*psat[grid->month]*loct->dlen[grid->month]*lTs/eK[grid->month];
			bb=eK[grid->month]*lue[grid->month]*loct->par[grid->month]/psat[grid->month];
			cc2=1.0+sqrt(1.0+bb);
```

```
                        cc3=1.0+sqrt(1.0+bb*exp(-1.0*eK[grid->month]*mass->lai[grid->month]));
                        gpp=cc1*log(cc2/cc3);
            }else{gpp=0.0;}
            return    gpp;
}
/*****    from    foliage,    maintenance    respiration    ****/
double Pchar::FoliMaintenRes(    Gchar *grid,        Gcon *loct, Pmas *mass)
{
            double    rfmt0,t0,rfm,ft;
            rfmt0=rmf/1000.0;      t0=15.0; /* specific    rate,    at    15 deg C */
            /*    temperature    dependence,    exponential    */
            ft=exp(log(qTf[grid->month])/10.0*(grid->tmp_sfc[grid->month]-t0));
            if(mass->fol>=0.0){
                        rfm=mass->fol*rfmt0*ft;
            }else{        rfm=0.0;}
            return(rfm);
}
/*****    from    foliage,    growth    and    construction    respiration    ****/
double Pchar::FoliGrowthRes( Gchar *grid,        Gcon *loct, Pmas *mass,    Pflx *flux)
{
            double rfg;
            /*    construction    cost    is    proportional    to    biomass    growth */
            rfg=rgf*flux->tpf[grid->month];
            return(rfg);
}
/****    from    stem and    branch,    maintenance    respiration    ****/
double Pchar::StemMaintenRes( Gchar *grid,        Gcon *loct, Pmas *mass)
{
            double    rfmt0,t0,rfm,ft;
            rfmt0=rmc/1000.0;      t0=15.0; /* specific    rate,    at    15 deg C    */
            /*    temperature    dependence,    exponential    */
            ft=exp(log(qTc[grid->month])/10.0*(grid->tmp_sfc[grid->month]-t0));
            if(mass->stm>=0.0){
                        rfm=mass->stm*rfmt0*ft;
            }else{rfm=0.0;}
            return(rfm);
}
/*****    from     stem and    branch,    growth    and    construction    respiration    ****/
double Pchar::StemGrowthRes( Gchar *grid,        Gcon *loct, Pmas *mass,    Pflx *flux)
{
            double rcg;
            /*    construction    cost    is    proportional    to    biomass    growth */
            rcg=rgc*flux->tpc[grid->month];    return(rcg);
}
/****    from    root,    maintenance    respiration      ****/
double Pchar::RootMaintenRes( Gchar *grid,        Gcon *loct, Pmas *mass)
{
            double    rfmt0,t0,rfm,ft;
            rfmt0=rmr/1000.0;      t0=15.0; /* specific    rate,    at    15 deg C    */
            /*    temperature    dependence,    exponential    */
            ft=exp(log(qTr[grid->month])/10.0*(grid->tmp_sfc[grid->month]-t0));
            if(mass->rot>=0.0){
                        rfm=mass->rot*rfmt0*ft;
            }else{rfm=0.0;}return(rfm);
}
/*****    from    root,    growth    and    construction    respiration    ****/
double Pchar::RootGrowthRes( Gchar *grid,        Gcon *loct,Pmas *mass,    Pflx *flux)
{
            double rrg;
            /*    construction    cost    is    proportional    to    biomass    growth */
            rrg= rgr * flux->tpr[grid->month];    return(rrg);
}
void Pchar::DoLitterFall(Gchar *grid, Gcon *loct, Pmas *mass, Pflx *flux)
{
            double area_frac;
```

```
		if(phototype==3){
			area_frac=loct->C3ptn;
		}else if(phototype==4){
			area_frac=loct->C4ptn;
			if(area_frac == 0){ mass->fol = 0; mass->stm=0;}
		}else area_frac =0;
		double nn=(double)grid->mm[grid->month];
		if(grid->vegNo == 2 && season[grid->month] == 3){
			/*	litter	*/
			flux->lf[grid->month] = dcd * mass->fol;	/*	leaf-shedding	*/
			flux->lc[grid->month] = dcd * mass->stm;
			flux->lr[grid->month] = nn* FromRoot(grid, loct, mass);
			/*	litter	fall	*/
			mass->fol -= flux->lf[grid->month];
			mass->stm -= flux->lc[grid->month];
			mass->rot -= flux->lr[grid->month]*area_frac;
		}else if(grid->vegNo == 2 && season[grid->month - 1] == 3){
			/*	litter	*/
			flux->lf[grid->month]= mass->fol;	/*	leaf-shedding	*/
			flux->lc[grid->month]= mass->stm;
			flux->lr[grid->month]=nn* FromRoot(grid, loct, mass);
			/*	litter	fall	*/
			mass->fol -= flux->lf[grid->month];
			mass->stm -= flux->lc[grid->month];
			mass->rot -= flux->lr[grid->month]*area_frac;
		}else if(grid->vegNo == 2 && season[grid->month] == 4){
			/*	litter	*/
			flux->lf[grid->month]= m_dLFProp * dcd * mass->fol;	/*	leaf-shedding	*/
			flux->lc[grid->month]= m_dLFProp * dcd * mass->stm;
			flux->lr[grid->month]=nn* FromRoot(grid, loct, mass);
			/*	litter	fall	*/
			mass->fol -= flux->lf[grid->month];
			mass->stm -= flux->lc[grid->month];
			mass->rot -= flux->lr[grid->month]*area_frac;
		}else {
			/*	litter	*/
			flux->lf[grid->month]=nn* FromFoliage(grid, loct, mass);
			flux->lc[grid->month]=nn* FromStem(grid, loct, mass);
			flux->lr[grid->month]=nn* FromRoot(grid, loct, mass);
			/*	litter	fall	*/
			mass->fol -= flux->lf[grid->month]*area_frac;
			mass->stm -= flux->lc[grid->month]*area_frac;
			mass->rot -= flux->lr[grid->month]*area_frac;
		}
}
```

```
// Pflx.h: interface for the Pchar class.
#if !defined(AFX_PFLX_H__A6700887_4D77_4B96_9736_5E51F44CCFCF__INCLUDED_)
#define AFX_PFLX_H__A6700887_4D77_4B96_9736_5E51F44CCFCF__INCLUDED_
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
class Bmas;
class Pflx
{
public:
        double      *gpp; /*  gross  primary  production */
        double      *spp; /*  net primary production    */
        double      *epp; /*  net primary production    */
        double      *npp; /*  net primary production    */
        double      *fnpp;
        double      *cnpp;
        double      *rnpp;
        double      *tpf;  /* translocation  of  photosynthate  to  foliage */
        double      *tpc;  /* translocation  of  photosynthate  to  stem */
        double      *tpr;  /* translocation  of  photosynthate  to  root */
        double      *tpp;  /* translocation  of  photosynthate  to  root */
        double      *rp; /*  plant  respiration,  =rpm+rpg */
        double      *rpg; /*  plant  growth respiration  */
        double      *rpm; /*  plant  maintenance  respiration  */
        double      *rfg; /*  foliage  growth respiration  */
        double      *rfm; /*  foliage  maintenance  respiration  */
        double      *rcg; /*  stem and branch growth respiration  */
        double      *rcm; /*  stem and branch  maintenance  respiration  */
        double      *rrg; /*  root growth respiration    */
        double      *rrm; /*  root  maintenance  respiration  */
        double      *lf; /*  foliage litterfall  */
        double      *lc; /*  stem and branch litterfall  */
        double      *lr; /*  root litterfall  */
        double      *lL; /*  total litterfall  */
        double      *hvst;  /*  harvest  of  crops  */
public:
        void reallocation_survival(Gchar *grid, Pchar  *pchar, Pmas *mass);
        void allocation(Gchar *grid, Pchar  *pchar,  Pmas *mass);
        void afterdeal( Gchar *grid,  Gcon *loct, Pchar  *pchar,  Pmas *mass);
        void beforedeal( Gchar *grid, Gcon *loct, Pchar  *pchar, Pmas *mass);
        void plant_flux_zero(long month);
public:
        void OtherAlloc(Gchar *grid, Pchar  *pchar, Pmas *mass);
        void MongoAlloc(Gchar *grid, Pchar  *pchar, Pmas *mass);
        Pflx();
        virtual ~Pflx();
};
#endif

// Pflx.cpp: interface for the Pchar class.
#include "stdafx.h"
#include "SimCycleMFC_OP.h"
#include "Pflx.h"
#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif
Pflx::Pflx()
{
        gpp = (double*)calloc(DAYNUM, sizeof(double));
        spp = (double*)calloc(DAYNUM, sizeof(double));
        epp = (double*)calloc(DAYNUM, sizeof(double));
```

```
		npp = (double*)calloc(DAYNUM, sizeof(double));
		fnpp = (double*)calloc(DAYNUM, sizeof(double));
		cnpp = (double*)calloc(DAYNUM, sizeof(double));
		rnpp = (double*)calloc(DAYNUM, sizeof(double));
		tpf = (double*)calloc(DAYNUM, sizeof(double));
		tpc = (double*)calloc(DAYNUM, sizeof(double));
		tpr = (double*)calloc(DAYNUM, sizeof(double));
		tpp = (double*)calloc(DAYNUM, sizeof(double));
		rp = (double*)calloc(DAYNUM, sizeof(double));
		rpg = (double*)calloc(DAYNUM, sizeof(double));
		rpm = (double*)calloc(DAYNUM, sizeof(double));
		rfg = (double*)calloc(DAYNUM, sizeof(double));
		rfm = (double*)calloc(DAYNUM, sizeof(double));
		rcg = (double*)calloc(DAYNUM, sizeof(double));
		rcm = (double*)calloc(DAYNUM, sizeof(double));
		rrg = (double*)calloc(DAYNUM, sizeof(double));
		rrm = (double*)calloc(DAYNUM, sizeof(double));
		lf = (double*)calloc(DAYNUM, sizeof(double));
		lc = (double*)calloc(DAYNUM, sizeof(double));
		lr = (double*)calloc(DAYNUM, sizeof(double));
		lL = (double*)calloc(DAYNUM, sizeof(double));
		hvst = (double*)calloc(DAYNUM, sizeof(double));
}
Pflx::~Pflx()
{
		free(gpp);free(spp);free(epp);free(npp);free(fnpp);
		free(cnpp);free(rnpp);free(tpf);free(tpc);free(tpr);
		free(tpp);free(rp);free(rpg);free(rpm);free(rfg);
		free(rfm);free(rcg);free(rcm);free(rrg);free(rrm);
		free(lf);free(lc);free(lr);free(lL);free(hvst);
}
/********	dealings	before	calculating	carbon	budget	**********/
void Pflx::beforedeal(Gchar *grid, Gcon *loct,	Pchar	*pchar,	Pmas *mass)
{		plant_flux_zero(grid->month);}
/********	dealings	after	calculating	carbon	budget	**********/
void Pflx::afterdeal(Gchar *grid, Gcon *loct, Pchar	*pchar, Pmas *mass)
{
		mass->lai[grid->month]= pchar->lai_mass(mass);
		rpg[grid->month]=rfg[grid->month]+rcg[grid->month]+rrg[grid->month];
		rp[grid->month]=rpm[grid->month]+rpg[grid->month];
		spp[grid->month]=gpp[grid->month]-rfm[grid->month]-rfg[grid->month];
		npp[grid->month]=gpp[grid->month]-rpm[grid->month]-rpg[grid->month];
		lL[grid->month]=lf[grid->month]+lc[grid->month]+lr[grid->month];
		/*	monthly	mass	values	*/
		if(mass->rot == 0) mass->stm =0;
		mass->mfol[grid->month]=mass->fol;
		mass->mstm[grid->month]=mass->stm;
		mass->mrot[grid->month]=mass->rot;
		mass->plant[grid->month]=mass->fol+mass->stm+mass->rot;
}
/***********	allocation	of	photosynthate	**************/
void Pflx::allocation(Gchar *grid, Pchar	*pchar, Pmas *mass)
{
		if(grid->vegNo == 2)// sparse glass land
				MongoAlloc(grid, pchar, mass);
		else OtherAlloc(grid, pchar, mass);
}
void Pflx::MongoAlloc(Gchar *grid, Pchar	*pchar, Pmas *mass)
{
		double	aaa, bbb, cc1,ccc, ddd;
		double	alloc_f,alloc_c,alloc_r;	/*	allocation	ratios	*/
		if(epp[grid->month]>0.0){	/*	during	growing-period	*/
				/*	allocation	ratios	of	EPP */
				if(mass->lai[grid->month] > pchar->opt_lai[grid->month]){
						/*	if	holding	LAI	is	greater	than	the	optimam	one	*/
```

```
                ccc= pchar->alloc_ass *1.0;// around 0.24
                alloc_f=ccc;
                alloc_c=(1.0-ccc) * pchar->alloc_abg; // 0.05
                alloc_r=(1.0-ccc) * (1.0-pchar->alloc_abg);
        }else if(mass->lai[grid->month] <= pchar->opt_lai[grid->month]){       /*  */
                /* if holding LAI is smaller than the optimam one */
                aaa=(pchar->opt_lai[grid->month] - mass->lai[grid->month])*100.0*2.0/2.2/pchar->sla;
                bbb=epp[grid->month] * pchar->alloc_ass;
                /* allocate photosyntahte to foliage to attain the optimum one */
                if(grid->vegNo == 2 ){
                        if(aaa<=bbb){
                                ccc=pchar->alloc_ass;
                        }else if(aaa>bbb){
                                cc1=aaa/bbb;
                if(grid->month < 8)ccc=((pchar->alloc_ass*cc1)<0.43)?(pchar->alloc_ass*cc1):0.43;
                else ccc= ((pchar->alloc_ass*cc1)<0.3)?(pchar->alloc_ass*cc1):0.3;
                        }
                        if(pchar->season[grid->month]==0) ddd=0.01;
                        else ddd =1;
                }else{
                        if(pchar->season[grid->month]==0) ddd=0.01;
                        else ddd =.7;
                }
                alloc_f = ccc*ddd;
                alloc_c = (1.0-ccc*ddd)*pchar->alloc_abg;
                alloc_r = (1.0-ccc*ddd)*(1.0-pchar->alloc_abg);
        }
        /* monthly translocation fluxes */
        tpp[grid->month]=(alloc_f+alloc_c+alloc_r)*epp[grid->month];
        tpf[grid->month]=alloc_f*epp[grid->month];
        tpc[grid->month]=alloc_c*epp[grid->month];
        tpr[grid->month]=alloc_r*epp[grid->month];
}else if(epp[grid->month]<=0.0){    /* during NON growing-period */
        /* allocation ratios of GPP, not EPP */
        alloc_f=pchar->alloc_ass;                       // around 0.24
        alloc_c=(1.0-pchar->alloc_ass)*pchar->alloc_abg;
        alloc_r=(1.0-pchar->alloc_ass)*(1.0-pchar->alloc_abg);
        /* monthly translocation fluxes */
        tpp[grid->month]=(alloc_f+alloc_c+alloc_r)*epp[grid->month];
        tpf[grid->month]=alloc_f*gpp[grid->month] - rfm[grid->month];
        tpc[grid->month]=alloc_c*gpp[grid->month] - rcm[grid->month];
        tpr[grid->month]=alloc_r*gpp[grid->month] - rrm[grid->month];
}
/* monthly partitioning ratios */
pchar->malloc_f[grid->month]=alloc_f;
pchar->malloc_c[grid->month]=alloc_c;
pchar->malloc_r[grid->month]=alloc_r;
}

void Pflx::OtherAlloc(Gchar *grid, Pchar    *pchar, Pmas *mass)
{
        double  aaa, bbb, cc1,ccc, ddd;
        double  alloc_f,alloc_c,alloc_r;    /* allocation ratios */
        if(epp[grid->month]>0.0){    /* during growing-period */
                /* allocation ratios of EPP */
                if(mass->lai[grid->month] > pchar->opt_lai[grid->month]){
                        /* if holding LAI is greater than the optimam one */
                        ccc= pchar->alloc_ass *1.0;// around 0.24
                        alloc_f=ccc;
                        alloc_c=(1.0-ccc) * pchar->alloc_abg; // 0.05
                        alloc_r=(1.0-ccc) * (1.0-pchar->alloc_abg);
                }else if(mass->lai[grid->month] <= pchar->opt_lai[grid->month]){       /*  */
                        /* if holding LAI is smaller than the optimam one */
                        aaa=(pchar->opt_lai[grid->month] - mass->lai[grid->month])*100.0*2.0/2.2/pchar->sla;
                        bbb=epp[grid->month] * pchar->alloc_ass;
```

```
                        /* allocate photosyntahte to foliage to attain the optimum one */
                        if(aaa<=bbb){
                                ccc=pchar->alloc_ass;
                        }else if(aaa>bbb){
                                cc1=aaa/bbb;
                                ccc=((pchar->alloc_ass*cc1)<0.5)?(pchar->alloc_ass*cc1):0.5;
                        }
                        if(pchar->season[grid->month]==0){
                                ddd=0.7;
                        }else{ddd=1.0;}
                        alloc_f = ccc*ddd;
                        alloc_c = (1.0-ccc*ddd)*pchar->alloc_abg;
                        alloc_r = (1.0-ccc*ddd)*(1.0-pchar->alloc_abg);
                }
                /* monthly translocation fluxes */
                tpp[grid->month]=(alloc_f+alloc_c+alloc_r)*epp[grid->month];
                tpf[grid->month]=alloc_f*epp[grid->month];
                tpc[grid->month]=alloc_c*epp[grid->month];
                tpr[grid->month]=alloc_r*epp[grid->month];
        }else if(epp[grid->month]<=0.0){ /* during NON growing-period */
                /* allocation ratios of GPP, not EPP */
                alloc_f=pchar->alloc_ass;                       // around 0.24
                alloc_c=(1.0-pchar->alloc_ass)*pchar->alloc_abg;
                alloc_r=(1.0-pchar->alloc_ass)*(1.0-pchar->alloc_abg);
                /* monthly translocation fluxes */
                tpp[grid->month]=(alloc_f+alloc_c+alloc_r)*epp[grid->month];
                tpf[grid->month]=alloc_f*gpp[grid->month] - rfm[grid->month];
                tpc[grid->month]=alloc_c*gpp[grid->month] - rcm[grid->month];
                tpr[grid->month]=alloc_r*gpp[grid->month] - rrm[grid->month];
        }
        /* monthly partitioning ratios */
        pchar->malloc_f[grid->month]=alloc_f;
        pchar->malloc_c[grid->month]=alloc_c;
        pchar->malloc_r[grid->month]=alloc_r;
}

/************* plant-internal mass re-adjustment *************/
void Pflx::reallocation_survival(Gchar *grid, Pchar *pchar, Pmas *mass)
{
        double ral_cap_stf,ral_cap_rtf;
        double aaa, bbb,ccc;
        double ral_stf, ral_rtf;
        switch(grid->vegNo){
                case 4:
                        ral_cap_stf=0.1;ral_cap_rtf=0.2;break;
                default:
                        ral_cap_stf=0.2;ral_cap_rtf=0.4;break;
        }
        /* to fliage: critical reallocation for survival */
        aaa = 0.1*100.0*2.0/2.2/pchar->sla;
        if(mass->fol< aaa){
                bbb=mass->stm*ral_cap_stf;
                ccc=mass->rot*ral_cap_rtf;
                ral_stf=aaa*pchar->alloc_abg*(bbb/aaa)/(0.5+(bbb/aaa));
                ral_rtf=aaa*(1.0-pchar->alloc_abg)*(ccc/aaa)/(0.5+(ccc/aaa));
                mass->fol+=ral_stf+ral_rtf;
                mass->stm-=ral_stf;
                mass->rot-=ral_rtf;
        }
        /* to fliage, improve production */
        aaa=(pchar->opt_lai[grid->month]-mass->lai[grid->month])*100.0*2.0/2.2/pchar->sla;
        if(aaa>0.0){
                bbb=mass->stm*ral_cap_stf;
                ccc=mass->rot*ral_cap_rtf;
                ral_stf=aaa*pchar->alloc_abg*(bbb/aaa)/(1.0+(bbb/aaa));
```

```
                ral_rtf=aaa*(1.0-pchar->alloc_abg)*(ccc/aaa)/(1.0+(ccc/aaa));
                mass->fol+=ral_stf+ral_rtf;mass->stm-=ral_stf;mass->rot-=ral_rtf;
        }
        mass->lai[grid->month]= pchar->lai_mass(mass);
        /* to stem and branch */
        if(mass->stm<=0.0){
                if(mass->rot>0.0){
                        aaa=mass->rot*pchar->alloc_abg;
                        mass->rot-=aaa;mass->stm+=aaa;
                }
        }
        /* to root */
        if(mass->rot<=0.0){
                if(mass->stm>0.0){
                        aaa=mass->stm*(1.0-pchar->alloc_abg);
                        mass->rot+=aaa;mass->stm-=aaa;
                }
        }
}
void Pflx::plant_flux_zero(long month)
{
        gpp[month]=0.0; epp[month]=0.0;   spp[month]=0.0; npp[month]=0.0;
        fnpp[month]=0.0; cnpp[month]=0.0; rnpp[month]=0.0;
        rfm[month]=0.0;         rcm[month]=0.0; rrm[month]=0.0; rpm[month]=0.0;
        rfg[month]=0.0;         rcg[month]=0.0; rrg[month]=0.0; rpg[month]=0.0;   rp[month]=0.0;
        lf[month]=0.0;lc[month]=0.0; lr[month]=0.0; lL[month]=0.0;
        tpp[month]=0.0;         tpf[month]=0.0; tpc[month]=0.0; tpr[month]=0.0;   hvst[month]=0.0;
}
```

```
// Pmas.h: interface for the Pmas class.
#if !defined(AFX_PMAS_H__369260AB_8C42_4AC3_8AF7_E4F54AA34E0E__INCLUDED_)
#define AFX_PMAS_H__369260AB_8C42_4AC3_8AF7_E4F54AA34E0E__INCLUDED_
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
class Pmas
{
public:
		double	fol;		/*	foliage	mass, Mg C ha-1	*/
		double	*mfol;		/*	monthly	foliage	mass, Mg C ha-1	*/
		double	*lai;		/*	monthly	leaf	area	index(LAI), m2 m-2	*/
		double	stm;		/*	stem and branch mass, Mg C ha-1	*/
		double	*mstm;		/*	monthly	stem and branch mass, Mg C ha-1 */
		double	rot;		/*	root	mass, Mg C ha-1	*/
		double	*mrot;		/* monthly root	mass, Mg C ha-1	*/
		double	fol_p, stm_p, rot_p;
		double	*plant;		/*	monthly	plant	mass	*/
public:
		Pmas();
		virtual ~Pmas();
};
#endif

// Pmas.cpp: implementation of the Pmas class.
#include "stdafx.h"
#include "SimCycleMFC_OP.h"
#include "Pmas.h"
#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif
Pmas::Pmas()
{
		mfol = (double*)calloc(DAYNUM, sizeof(double));
		lai = (double*)calloc(DAYNUM, sizeof(double));
		mstm = (double*)calloc(DAYNUM, sizeof(double));
		mrot = (double*)calloc(DAYNUM, sizeof(double));
		plant = (double*)calloc(DAYNUM, sizeof(double));
}
Pmas::~Pmas()
{free(mfol); free(lai);free(mstm);free(mrot);free(plant);}
```

```
// PrintData.h: interface for the PrintData class.
#if !defined(AFX_PRINTDATA_H__E2D81C20_5421_4413_996F_76406CA154BE__INCLUDED_)
#define AFX_PRINTDATA_H__E2D81C20_5421_4413_996F_76406CA154BE__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
class PrintData
{
public:
		FILE *m_fpResult;
		void PrintElementInt(int iEntry);
		void PrintElement(double dEntry);
		void PrintFlux(Cflx *flux, int f);
		void PrintFluxC3(Cflux *flux, int f);
		void PrintFluxSoil(Cflx *flux, int f);
		void PrintBmas(Bmas *mass, int f);
		void PrintLocation(Gchar *grid, Gcon *loct, Echar	*ecochar, int f);
		void PrintInFile(	Gchar *grid,	Gcon	*loct,		Echar	*ecochar,
								Bmas	*mass,	Cflx	*flux,	FILE *result);
		PrintData(int iFlag);
		virtual ~PrintData();
};
#endif

// PrintData.cpp: implementation of the PrintData class.
#include "stdafx.h"
#include "SimCycleMFC_OP.h"
#include "PrintData.h"
#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif
PrintData::PrintData(int iFlag)
{}
PrintData::~PrintData()
{}
void PrintData::PrintInFile(	Gchar *grid,	Gcon	*loct,		Echar	*ecochar,
								Bmas	*mass,	Cflx	*flux,	FILE *result)
{
		m_fpResult = result;
		PrintElement(grid->lat); PrintElement(grid->lon);PrintElementInt(grid->time);PrintElementInt(grid->nnn);
		PrintElement(grid->area);PrintElement(loct->C4ptn);PrintElement(grid->tmp_sfc_am);
		PrintElement(grid->xprate_sfc_ann); fprintf(result,"\n");
		for(int f=0;f<DAYNUM;f++){
			PrintLocation(grid, loct, ecochar, f);
			PrintBmas(mass, f);
			PrintFlux(flux, f);
			PrintFluxC3(flux, f);
			PrintFluxSoil(flux, f);
			fprintf(result,"\n");
		}
		fprintf(result,"\n");
}

void PrintData::PrintLocation(Gchar *grid, Gcon *loct, Echar	*ecochar, int f)
{
		PrintElement(grid->aCO2[f]);PrintElement(loct->top_rad[f]); PrintElement(loct->gl_rad[f]);
		PrintElement(loct->rad_net_p[f]);PrintElement(loct->rad_net_p[f]+loct->rad_net_g[f]);
		PrintElement(loct->pm_evp[f]);PrintElement(loct->par[f]);PrintElement(loct->fapar[f]);
		PrintElement(loct->apar[f]);PrintElement(grid->xprate_sfc[f]);PrintElement(loct->evpr[f]);
		PrintElement(loct->trspr[f]);PrintElement(loct->ro2[f]);PrintElement(loct->msw30[f]);
```

```
		PrintElement(loct->msww[f]);
}
void PrintData::PrintBmas(Bmas *mass, int f)
{
		PrintElement((mass->plant)->lai[f]); PrintElement((mass->plant)->mfol[f]);
		PrintElement((mass->plant)->mstm[f]);PrintElement((mass->plant)->mrot[f]);
		PrintElement((mass->c3)->lai[f]);PrintElement((mass->c3)->mfol[f]);
		PrintElement((mass->c3)->mstm[f]); PrintElement((mass->c3)->mrot[f]);
		PrintElement((mass->c4)->lai[f]);PrintElement((mass->c4)->mfol[f]);
		PrintElement((mass->c4)->mstm[f]); PrintElement((mass->c4)->mrot[f]);
		PrintElement((mass->soil)->mltr[f]); PrintElement((mass->soil)->mmsl[f]);
}
void PrintData::PrintFlux(Cflx *flux, int f)
{
		PrintElement((flux->plant)->gpp[f]); PrintElement((flux->plant)->spp[f]);
		PrintElement((flux->plant)->epp[f]); PrintElement((flux->plant)->npp[f]);
		PrintElement((flux->plant)->tpf[f]);  PrintElement((flux->plant)->tpc[f]);
		PrintElement((flux->plant)->tpr[f]);  PrintElement((flux->plant)->rfm[f]);
		PrintElement((flux->plant)->rcm[f]); PrintElement((flux->plant)->rrm[f]);
		PrintElement((flux->plant)->rfg[f]);  PrintElement((flux->plant)->rcg[f]);
		PrintElement((flux->plant)->rrg[f]);  PrintElement((flux->plant)->lf[f]);
		PrintElement((flux->plant)->lc[f]);   PrintElement((flux->plant)->lr[f]);
}
void PrintData::PrintFluxC3(Cflx *flux, int f)
{
		PrintElement((flux->c3)->gpp[f]);    PrintElement((flux->c3)->spp[f]);
		PrintElement((flux->c3)->epp[f]);    PrintElement((flux->c3)->npp[f]);
		PrintElement((flux->c3)->tpf[f]);    PrintElement((flux->c3)->tpc[f]);
		PrintElement((flux->c3)->tpr[f]);    PrintElement((flux->c3)->rfm[f]);
		PrintElement((flux->c3)->rcm[f]);    PrintElement((flux->c3)->rrm[f]);
		PrintElement((flux->c3)->rfg[f]);    PrintElement((flux->c3)->rcg[f]);
		PrintElement((flux->c3)->rrg[f]);    PrintElement((flux->c3)->lf[f]);
		PrintElement((flux->c3)->lc[f]);     PrintElement((flux->c3)->lr[f]);
}
void PrintData::PrintFluxSoil(Cflx *flux, int f)
{
		PrintElement((flux->plant)->fnpp[f]);PrintElement((flux->plant)->cnpp[f]);
		PrintElement((flux->plant)->rnpp[f]);PrintElement((flux->soil)->rl[f]);
		PrintElement((flux->soil)->rh[f]);PrintElement((flux->soil)->rS[f]);
		PrintElement(flux->nep[f]);PrintElement(flux->ncb[f]);PrintElement((flux->soil)->sf[f]);
}
void PrintData::PrintElement(double dEntry)
{		fprintf(m_fpResult,"%.3lf\t", dEntry);}

void PrintData::PrintElementInt(int iEntry)
{		fprintf(m_fpResult,"%d\t", iEntry);}
```

```
// Schar.h: interface for the Schar class.
#if !defined(AFX_SCHAR_H__D9AB45A6_0899_46CC_BAD2_6F885DEC3019__INCLUDED_)
#define AFX_SCHAR_H__D9AB45A6_0899_46CC_BAD2_6F885DEC3019__INCLUDED_
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
class Schar
{
public:
		double	albedo0, *albedo;	/* reflectivity, or albedo */
		double	rl; /* specific respiration rate at 15degC, mg C g dm-1 day-1 */
		double	rh; /* specific respiration rate at 15degC, mg C g dm-1 day-1 */
		double	qTl, qTh; /* emperature dependence, dimensionless */
		double	kml, kmh; /* moisture dependence, fraction of soil water */
		double	kmsl, kmsh; /* moisture dependence, fraction of soil water */
		double	me; /* mineral soil formation ratio to litter decomposition, fraction */
public:
		void SetParameterSoil(Gchar *grid);
		Schar();
		virtual ~Schar();
};
#endif
// Schar.cpp: interface for the Schar class.
#include "stdafx.h"
#include "SimCycleMFC_OP.h"
#include "Schar.h"
#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif
Schar::Schar()
{		albedo = (double*)calloc(DAYNUM, sizeof(double));}

Schar::~Schar()
{		free(albedo);}

void Schar::SetParameterSoil(Gchar *grid)
{
		albedo0=0.05; rl= 1.45; rh= 0.127;qTl= 2.0; qTh= 2.0;
		kml= 0.20; kmh= 0.11;kmsl= 0.20; kmsh= 0.10;me= 1.20;
}
```

```
// Sflx.h: interface for the Schar class.
#if !defined(AFX_SFLX_H__59AE0F82_8710_47B0_B66F_3CCED13C6D49__INCLUDED_)
#define AFX_SFLX_H__59AE0F82_8710_47B0_B66F_3CCED13C6D49__INCLUDED_
#if _MSC_VER > 1000
#pragma once
#endif
class Smas;
class Sflx
{
public:
		double		*lL; /*	litter input	*/
		double		*rl; /*	litter decomposition	*/
		double		*rh; /*	mineral soil and humus decomposition	*/
		double		*rS; /*	total decomposition	*/
		double		*sf; /*	humus formation	*/
public:
		double HumusFormation(Gchar *grid, Gcon *loct, Schar *soil, Smas *mass);
		double MineralRes( Gchar *grid, Gcon *loct, Schar	*soil, Smas *mass);
		double LitterRes( Gchar *grid,	Gcon *loct, Schar *soil, Smas *mass);
		void DoSoilProcess(Gchar *grid, Gcon *loct, Schar *schar, Smas *mass);
public:
		Sflx();virtual ~Sflx();
};
#endif

// Sflx.cpp: implementation of the Sflx class.
#include "stdafx.h"
#include "SimCycleMFC_OP.h"
#include "Sflx.h"
#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif
Sflx::Sflx()
{
		lL = (double*)calloc(DAYNUM, sizeof(double));rl = (double*)calloc(DAYNUM, sizeof(double));
		rh = (double*)calloc(DAYNUM, sizeof(double));rS = (double*)calloc(DAYNUM, sizeof(double));
		sf = (double*)calloc(DAYNUM, sizeof(double));
}
Sflx::~Sflx()
{		free(lL);free(rl);free(rh);free(rS);free(sf);}
/******	yearly	prcesses	in	belowground	******/
void Sflx::DoSoilProcess(	Gchar *grid,	Gcon	*loct, Schar *schar,	Smas	*mass)
{
		double	nn;
		nn=(double)grid->mm[grid->month];
		if(grid->vegNo==0||grid->vegNo==97||grid->vegNo==98||grid->vegNo==99||grid->vegNo==100){
			/*	no	soil	carbon	in	bare lands	*/
			rl[grid->month]=0.0;rh[grid->month]=0.0;sf[grid->month]=0.0;
			mass->ltr=0.0;mass->msl=0.0;
		}else{
			/*	soil	respiration	of	litter	layer */
			rl[grid->month] = nn* LitterRes(grid,	loct,	schar,	mass );
			/*	soil	respiration	of	mineral	soil	and	humus */
			rh[grid->month] = nn* MineralRes( grid,	loct,	schar,	mass);
			/*	soil	decomposition	from	upper	litter to	lower	mineral	soil */
			sf[grid->month]= HumusFormation( grid,	loct,	schar,	mass );
			/*	mass	balance	*/
			mass->ltr+=lL[grid->month]-rl[grid->month]-sf[grid->month];
			mass->msl+=sf[grid->month]-rh[grid->month];
			mass->ltr=(mass->ltr>=0.0)?mass->ltr:0.0;
```

```
                        mass->msl=(mass->msl>=0.0)?mass->msl:0.0;
            }
            /* total soil respiration */
            rS[grid->month]=rl[grid->month]+rh[grid->month];
            /* monthly values */
            mass->mltr[grid->month]= mass->ltr;
            mass->mmsl[grid->month]= mass->msl;
            mass->soil[grid->month]= mass->ltr + mass->msl;
}
/******          soil respiration of litter layer , frl() ******/
double Sflx::LitterRes( Gchar *grid,      Gcon *loct, Schar   *soil,      Smas *mass)
{
            double rlto,rl,ftl,fwl, fal, fsm;
            rlto=soil->rl/1000.0;
            /* temperature effect, exponential */
            if(grid->tmp10_soil[grid->month]>-20.0){
                        ftl=0.4+0.6*exp(308.56*(1.0/56.02-1.0/(grid->tmp10_soil[grid->month]+46.02)));
            }else{ftl=0.2;}
            /* soil moisture effect, saturating */
            fwl=0.8*loct->sw30/(soil->kml*grid->whc30+loct->sw30)+0.2;
            /*  soil apparence effect  */
            fal=0.4*loct->soil_appr30*(1.0*soil->kmsl)/(soil->kmsl+loct->soil_appr30)+0.6;
            fsm=(fwl>fal)?fal:fwl;
            rl=mass->ltr*rlto*ftl*fsm;
            /* in case of too much emission, in order to avoid negative mass value */
            if((mass->ltr-rl*(1.0+soil->me))<0.0){rl=mass->ltr/(1.0+soil->me);}
            return(rl);
}
/******          soil respiration of mineral soil layer  ******/
double Sflx::MineralRes( Gchar *grid,      Gcon *loct, Schar   *soil,      Smas *mass)
{
            double rhto,rh,fth,fwh,fah, fsm;
            rhto=soil->rh/1000.0;
            /* temperature effect, exponential */
            if(grid->tmp200_soil[grid->month]>-20.0){
                        fth=0.4+0.6*exp(308.56*(1.0/56.02-1.0/(grid->tmp200_soil[grid->month]+46.02)));
            }else{fth=0.2;}
            /* soil moisture effect, saturating */
            fwh= 0.4 + 0.6*loct->sww/(soil->kmh*(grid->whc-40)+loct->sww);
            /*  soil apparence effect  */
            fah=0.4*loct->soil_apprw*(1.0*soil->kmsh)/(soil->kmsh+loct->soil_apprw)+0.6;
            fsm=(fwh>fah)?fah:fwh;
            rh=mass->msl*rhto*fth*fsm;
            /* in case of too much emission, in order to avoid negative mass value */
            if((mass->msl-rh)<0.0){ rh=mass->msl;}
            return(rh);
}
/****      humus formation from litter to mineral soil ****/
double Sflx::HumusFormation(  Gchar *grid,      Gcon *loct, Schar   *soil,      Smas *mass)
{
            double sf;
            /* humus formation is assumed to proceed in paralell with litter respiration, proportionally */
            sf=soil->me*rl[grid->month];return(sf);
}
```

```
// Smas.h: interface for the Smas class.
#if !defined(AFX_SMAS_H__2BA7939E_E220_4CF4_A9E3_6F9624190780__INCLUDED_)
#define AFX_SMAS_H__2BA7939E_E220_4CF4_A9E3_6F9624190780__INCLUDED_
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
class Smas
{
public:
        double      ltr; /*  litter  mass    */
        double      *mltr; /*  monthly litter  mass    */
        double      msl; /*  mineral  soil  and  humus  mass  */
        double      *mmsl; /* monthly  mineral  soil  and  humus  mass  */
        double      *soil; /*  monthly  soil  mass  */
public:
        Smas();virtual ~Smas();
};
#endif

// Smas.cpp: implementation of the Smas class.
#include "stdafx.h"
#include "SimCycleMFC_OP.h"
#include "Smas.h"
#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif
Smas::Smas()
{
        mltr = (double*)calloc(DAYNUM, sizeof(double));
        mmsl = (double*)calloc(DAYNUM, sizeof(double));
        soil = (double*)calloc(DAYNUM, sizeof(double));
}
Smas::~Smas()
{       free(mltr);   free(mmsl); free(soil);}
```

```
// TransSimulation.h: interface for the TransSimulation class.
#if !defined(AFX_TRANSSIMULATION_H__22173AE1_AA58_4770_91A7_5C6031A138DC__INCLUDED_)
#define AFX_TRANSSIMULATION_H__22173AE1_AA58_4770_91A7_5C6031A138DC__INCLUDED_
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
class TransSimulation
{
public:
		CString m_sFilePath;
		bool m_bTrans;
		double m_dPreRate;
		double m_dTempRate;
		int m_iRoundYear;
		int m_iTSimYear;
		FILE *m_fpTransRes;
		FILE *m_fpTransResYear;
public:
		void MakeHeader();
		void PrintDataDouble(FILE *fp, double dData);
		void PrintDataInt(FILE *fp, int dData);
		void PrintData(Gchar *grid, Gcon *loct, Echar *ecochar, Bmas *mass, Cflx *flux);
		void FileClose();
		void MakeDir(CString DirName);
		void SetFilePath(CString sFilePath);
		void SetFile(CString sFilePath);
		TransSimulation();
		virtual ~TransSimulation();
};
#endif

// TransSimulation.cpp: implementation of the TransSimulation class.
#include "stdafx.h"
#include "SimCycleMFC_OP.h"
#include "TransSimulation.h"
#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif
TransSimulation::TransSimulation()
{		m_iRoundYear = 0;}
TransSimulation::~TransSimulation()
{		m_bTrans = true;}
void TransSimulation::SetFile(CString sFilePath)
{
		SetFilePath(sFilePath);
		m_fpTransRes =fopen(sFilePath + "TransResMonth.dat","wt");
		m_fpTransResYear =fopen(sFilePath + "TransResYear.dat","wt");
		MakeHeader();
}
void TransSimulation::SetFilePath(CString sFilePath)
{
		m_sFilePath = sFilePath;
		MakeDir(m_sFilePath);
		CString sTemp;
		sTemp.Format("%d", m_iRoundYear);
		sTemp += ".dat";
		m_sFilePath += sTemp;
}
void TransSimulation::MakeDir(CString DirName)
{
		CFileFind finder;
```

```
		BOOL bExist = finder.FindFile(DirName);
		finder.Close();
		if(!bExist){
			if ( !(CreateDirectory((LPCTSTR)DirName, NULL)))
			{if (	GetLastError()	== ERROR_CANNOT_COPY ) return;}
		}
}
void TransSimulation::FileClose()
{		fclose(m_fpTransRes);	fclose(m_fpTransResYear);}
void TransSimulation::PrintData(Gchar *grid, Gcon *loct, Echar *ecochar, Bmas *mass, Cflx *flux)
{
		double dRad = 0;double dPAR = 0;double dPrecip = 0;double dTemp2m =0;double dTempSfc = 0;
		double GPP=0;double NPP=0;double ANPP = 0;double BNPP = 0;double AR=0;double ARG=0;
		double ARM=0;double LF=0;double NEP=0;double A_AR = 0;double U_AR = 0;double HR =0 ;
		double dWE = 0;double dAB = 0;double dLeaf=0;double dStem=0;double dRoot=0;
		double dsww=0;double dsw30=0;double dEvepo=0;double dTransp=0;double dRO1=0;double dRO2=0;
		for(int i =0 ; i< 12 ; i++){
			PrintDataInt(m_fpTransRes, m_iRoundYear);
			PrintDataDouble(m_fpTransRes, grid->dswrf_sfc[i] ); dRad += grid->dswrf_sfc[i];
			PrintDataDouble(m_fpTransRes, loct->par[i] );	dPAR += loct->par[i];
			PrintDataDouble(m_fpTransRes, grid->xprate_sfc[i] ); dPrecip += grid->xprate_sfc[i];
			PrintDataDouble(m_fpTransRes, grid->tmp_2m[i] );
			if(i >= 3 && i <9)		dTemp2m += (grid->tmp_2m[i] );
			PrintDataDouble(m_fpTransRes, grid->tmp_sfc[i]); dTempSfc += (grid->tmp_sfc[i] );
			PrintDataDouble(m_fpTransRes, mass->plant->lai[i] );
			PrintDataDouble(m_fpTransRes, mass->plant->mfol[i] + mass->plant->mstm[i] +mass->plant->mrot[i] +
			mass->soil->mltr[i] + mass->soil->mmsl[i] );
			dWE = (mass->plant->mfol[7] + mass->plant->mstm[7] +mass->plant->mrot[7] +mass->soil->mltr[7] +
			mass->soil->mmsl[7]);
			PrintDataDouble(m_fpTransRes, mass->plant->mfol[i] +mass->plant->mstm[i]);
			dAB = (mass->plant->mfol[7] +mass->plant->mstm[7]);
			PrintDataDouble(m_fpTransRes, mass->plant->mrot[i] );
			dRoot = (mass->plant->mrot[9]);
			PrintDataDouble(m_fpTransRes, mass->soil->mltr[i] );
			PrintDataDouble(m_fpTransRes, mass->soil->mmsl[i] );
			PrintDataDouble(m_fpTransRes, flux->plant->gpp[i]	);
			GPP += (flux->plant->gpp[i]);
			PrintDataDouble(m_fpTransRes, flux->plant->tpf[i] +flux->plant->tpc[i] –
								flux->plant->rfg[i] - flux->plant->rcg[i]);
			ANPP +=	(flux->plant->tpf[i] +flux->plant->tpc[i] - flux->plant->rfg[i] - flux->plant->rcg[i]);
			PrintDataDouble(m_fpTransRes, flux->plant->npp[i] -
				(flux->plant->tpf[i] +flux->plant->tpc[i] - flux->plant->rfg[i] - flux->plant->rcg[i]));
			BNPP +=(flux->plant->npp[i] - (flux->plant->tpf[i] +flux->plant->tpc[i] -
								flux->plant->rfg[i] - flux->plant->rcg[i]));
			PrintDataDouble(m_fpTransRes, flux->plant->gpp[i] - flux->plant->npp[i]	);
			AR += (flux->plant->gpp[i] - flux->plant->npp[i]);
			PrintDataDouble(m_fpTransRes, flux->plant->rfm[i] +flux->plant->rcm[i] + flux->plant->rrm[i]);
			ARM +=(flux->plant->rfm[i] +flux->plant->rcm[i] + flux->plant->rrm[i]);
			PrintDataDouble(m_fpTransRes, flux->plant->rfg[i] +flux->plant->rcg[i] + flux->plant->rrg[i] );
			ARG +=(flux->plant->rfg[i] +flux->plant->rcg[i] + flux->plant->rrg[i]);
			PrintDataDouble(m_fpTransRes, flux->soil->rl[i]	);
			LF += (flux->soil->rl[i] );
			PrintDataDouble(m_fpTransRes, flux->soil->rS[i]	);
			HR += flux->soil->rS[i];
			PrintDataDouble(m_fpTransRes, flux->plant->npp[i]	);
			NPP += (flux->plant->npp[i]);
			PrintDataDouble(m_fpTransRes, flux->nep[i] );
			NEP += (flux->nep[i] );
			PrintDataDouble(m_fpTransRes, loct->msw30[i]	); dsw30 += loct->msw30[i]/12;
			PrintDataDouble(m_fpTransRes, loct->msww[i]	);	dsww += loct->msww[i]/12;
			PrintDataDouble(m_fpTransRes, loct->evpr[i]	); dEvepo +=	loct->evpr[i] ;
			PrintDataDouble(m_fpTransRes, loct->trspr[i]	); dTransp += loct->trspr[i];
			PrintDataDouble(m_fpTransRes, loct->ro1[i]	); dRO1 += loct->ro1[i];
			PrintDataDouble(m_fpTransRes, loct->ro2[i]	); dRO2 += loct->ro2[i];
			PrintDataDouble(m_fpTransRes, ecochar->c3->ci[i]	);
```

```
                        PrintDataDouble(m_fpTransRes, ecochar->c3->gs[i]      );
                        PrintDataDouble(m_fpTransRes, loct->vpd[i]      );
            }
            PrintDataInt(m_fpTransResYear, m_iRoundYear);PrintDataDouble(m_fpTransResYear, dRad    );
            PrintDataDouble(m_fpTransResYear, dPAR    );  PrintDataDouble(m_fpTransResYear, dPrecip );
            PrintDataDouble(m_fpTransResYear, dTemp2m/6);// 2m air
            PrintDataDouble(m_fpTransResYear, dTempSfc/12);// surface
            PrintDataDouble(m_fpTransResYear, mass->plant->lai[7] );   PrintDataDouble(m_fpTransResYear, dWE    );
            PrintDataDouble(m_fpTransResYear, dAB );       PrintDataDouble(m_fpTransResYear, dRoot    );
            PrintDataDouble(m_fpTransResYear, mass->soil->ltr );PrintDataDouble(m_fpTransResYear, mass->soil->msl    );
            PrintDataDouble(m_fpTransResYear, GPP );       PrintDataDouble(m_fpTransResYear, ANPP);
            PrintDataDouble(m_fpTransResYear, BNPP);       PrintDataDouble(m_fpTransResYear, AR);
            PrintDataDouble(m_fpTransResYear, ARM);PrintDataDouble(m_fpTransResYear, ARG);
            PrintDataDouble(m_fpTransResYear, LF);PrintDataDouble(m_fpTransResYear, HR );
            PrintDataDouble(m_fpTransResYear, NPP);PrintDataDouble(m_fpTransResYear, NEP );
            PrintDataDouble(m_fpTransResYear, dsw30 ); PrintDataDouble(m_fpTransResYear, dsww/250*100        );
            PrintDataDouble(m_fpTransResYear, dEvepo );PrintDataDouble(m_fpTransResYear, dTransp      );
            PrintDataDouble(m_fpTransResYear, dRO1 );PrintDataDouble(m_fpTransResYear, dRO2      );
            PrintDataDouble(m_fpTransResYear, 100*2.2*ANPP/(dEvepo+dTransp) );
            PrintDataDouble(m_fpTransResYear,grid->aCO2[7]);PrintDataDouble(m_fpTransResYear,ecochar->c3->m_dTest_ci[7]     );
            PrintDataDouble(m_fpTransResYear, ecochar->c3->gs[7] );fprintf(m_fpTransResYear,"\n");
}
void TransSimulation::PrintDataDouble(FILE *fp, double dData)
{           fprintf(fp, "%.3lf\t", dData);}
void TransSimulation::PrintDataInt(FILE *fp, int dData)
{           fprintf(fp, "%d\t", dData);}
void TransSimulation::MakeHeader()
{
            fprintf(m_fpTransRes, "Num\t");
            fprintf(m_fpTransRes, "Radi\tPAR\tPrecip.\tAirT\tSfcT\t");
            fprintf(m_fpTransRes, "LAI\tWE\tAB\tBB\tLitter\tMSL\t");
            fprintf(m_fpTransRes, "GPP\tANPP\tBNPP\tAR\tARM\tARG\tULF\tHR\tNPP\tNEP\t");
            fprintf(m_fpTransRes, "sw30\tsww\tEvepo\tTrans\t");
            fprintf(m_fpTransRes, "ro1\tro2\t");
            fprintf(m_fpTransRes, "\n");
            fprintf(m_fpTransResYear, "Num\t");
            fprintf(m_fpTransResYear, "Radi\tPAR\tPrecip.\tAirT\tSfcT\t");
            fprintf(m_fpTransResYear, "LAI\tWE\tAB\tBB\tLitter\tMSL\t");
            fprintf(m_fpTransResYear, "GPP\tANPP\tBNPP\tAR\tARM\tARG\tULF\tHR\tNPP\tNEP\t");
            fprintf(m_fpTransResYear, "sw30\tMSLW\tEV\tTR\t");
            fprintf(m_fpTransResYear, "ro1\tRunoff\t");
            fprintf(m_fpTransResYear, "WUE\tAtmCO2\tci\tgs\t");
            fprintf(m_fpTransResYear, "\n");
}
```

```
// Initiate.h: interface for the Initiate class.
#if !defined(AFX_INITIATE_H__32E99945_BE07_4156_B8AB_5E8BDFE20D32__INCLUDED_)
#define AFX_INITIATE_H__32E99945_BE07_4156_B8AB_5E8BDFE20D32__INCLUDED_
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
#include "SimCalculation.h"
class Initiate
{
public:
		int m_iReadNum;int m_iaDiv[g_iRRow];int m_iX;int m_iY;
		double m_topo[g_iRCol];double m_whc[g_iRCol];double m_wilting[g_iRCol];
		double m_tmp_sfc[g_iRCol][12];double m_tmp_2m[g_iRCol][12];double m_tmp10_soil[g_iRCol][12];
		double m_tmp200_soil[g_iRCol][12];double m_dswrf_sfc[g_iRCol][12];double m_tcdc_clm[g_iRCol][12];
		double m_xprate_sfc[g_iRCol][12];double m_spfh_2m[g_iRCol][12];double m_wind_10m[g_iRCol][12];
		SimCalculation *m_cSCal;
public:
		void InitGridTrans(Gchar *grid, TransSimulation *pTrans);
		void InitLocationCond(Gchar	*grid,	Gcon	*loct, Bmas	*mass, Echar *ecochar,
				SimCalculation *cSCal= NULL, TransSimulation *pTrans = NULL);
		void InitClimateCond(Gchar	*grid,	Gcon	*loct,	Bmas	*mass, Echar *ecochar);
		void InitVegCond(Gchar *grid, Gcon	*loct,	Bmas	*mass, Echar *ecochar);
		void Clear(Gchar *grid,	Gcon *loct,	Echar	*ecochar, Bmas *mass, Cflx *flux);
		void InitGrid(FILE *fp_s[12], Gchar *grid);
		void InitSim(Gchar *grid);
		Initiate();
		virtual ~Initiate();
private:
		void InitSoil(Gchar *grid, Bmas	*mass);void InitC3(Gchar *grid, Bmas	*mass);
};
#endif
#include "stdafx.h"
#include "SimCycleMFC_OP.h"
#include "Initiate.h"
#include <math.h>
#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif
// field data in KBU at 2003
double dPrecip[12] = {0.00 ,0.00 ,1.20 ,0.00 ,28.00 ,43.20 ,55.20 ,41.00 ,65.10 ,10.10 ,1.20 ,0.00 };
double dDwrn[12] = {86.65 ,123.93 ,185.98 ,202.57 ,253.47 ,264.61 ,252.62 ,244.02 ,192.14 ,147.83 ,98.31 ,73.35 };
double dAirTemp[12] = {-21.82 ,-14.45 ,-7.52 ,2.14 ,11.03 ,17.58 ,19.44 ,15.90 ,11.66 ,1.41 ,-10.73 ,-21.46	};
double dSufTemp[12] = {-17.87 ,-11.58 ,-5.23 ,5.15 ,14.66 ,21.36 ,22.78 ,19.71 ,14.15 ,2.46 ,-10.36 ,-18.29 };
double dST10[12] = {-21.95 ,-17.10 ,-9.13 ,6.31 ,14.30 ,20.47 ,21.90 ,20.11 ,14.01 ,2.89 ,-6.23 ,-16.12 };
double dST200[12] = {-5.91 ,-7.89 ,-7.36 ,-3.43 ,0.46 ,5.51 ,9.63 ,11.92 ,11.62 ,8.76 ,4.34 ,-1.36 };
double dWind[12] = {2.06 ,3.34 ,3.85 ,4.65 ,3.94 ,3.72 ,4.35 ,6.23 ,2.70 ,3.10 ,2.56 ,2.34 };
Initiate::Initiate()
{
		m_iReadNum = -1;
		m_iX = m_iY = -1;
		for(int i=0 ; i < g_iRCol ; i++){
			m_topo[i] = 0;m_whc[i] = 0;m_wilting[i] = 0;
			for(int j = 0 ; j < 12 ; j++){
				m_tmp_sfc[i][j] = 0;m_tmp_2m[i][j] = 0;m_tmp10_soil[i][j] = 0;
				m_tmp200_soil[i][j] = 0;m_dswrf_sfc[i][j] = 0;m_tcdc_clm[i][j] = 0;
				m_xprate_sfc[i][j] = 0;	m_spfh_2m[i][j] = 0;m_wind_10m[i][j] = 0;
			}
		}
}
Initiate::~Initiate()
{}
```

```
void Initiate::InitSim(Gchar *grid)
{
	grid->nnn=0;
	/*	Number of days for each	month	*/
	grid->mm[0]=31;	grid->mm[1]=28;	grid->mm[2]=31;	grid->mm[3]=30;
	grid->mm[4]=31;	grid->mm[5]=30;	grid->mm[6]=31;	grid->mm[7]=31;
	grid->mm[8]=30;	grid->mm[9]=31;	grid->mm[10]=30;	grid->mm[11]=31;
}
void Initiate::InitGrid(FILE *fp_s[], Gchar *grid)
{
	long	e;double	tmp_sfc,tmp_2m,tmp10_soil,tmp200_soil,dswrf_sfc,tcdc_clm;
	double	xprate_sfc,spfh_2m,wind_10m;double	geo_prop;
	grid->area= 55.6/6.0 * 55.6/6.0 * sin(((double)(grid->raw)/2.0+0.25)*PI/180.0);
	grid->lat= (90.0-1.0/12.0/2.0)-(1.0/12.0)*(double)grid->raw;
	grid->lon= (-180.0+1.0/12.0/2.0)+(1.0/12.0)*(double)grid->col;
	/**	input	soil properties **/
	fscanf(fp_s[0],"%lf",&geo_prop);
	geo_prop= (geo_prop>0.0)? geo_prop:0.0;
	grid->topo= geo_prop;	/*	topography	*/
	fscanf(fp_s[1],"%lf",&geo_prop);
	grid->whc=449.60;	/*	whc	whole	*/
	fscanf(fp_s[2],"%lf",&geo_prop);
	grid->wilt_point=geo_prop;	/*	wilting point	*/
	/**	input	monthly	climate	**/
	for(e=0;e<12;e++){
		grid->dswrf_sfc_a[e]=dDwrn[e];
		fscanf(fp_s[4],"%lf",&spfh_2m);	grid->spfh_2m_a[e]=spfh_2m;
		fscanf(fp_s[5],"%lf",&tcdc_clm);	grid->tcdc_clm_a[e]=tcdc_clm;
		grid->tmp10_soil_a[e]=dST10[e] +ZAT;
		grid->tmp200_soil_a[e]=dST200[e]+ZAT;
		grid->tmp_2m_a[e]=dAirTemp[e]+ZAT;
		grid->tmp_sfc_a[e]=dSufTemp[e]+ZAT;
		grid->wnd_10m_a[e]=dWind[e];
		grid->xprate_sfc_a[e] = dPrecip[e]/30;
	}
}
void Initiate::Clear(Gchar *grid,	Gcon *loct,	Echar	*ecochar,	Bmas *mass, Cflx *flux)
{
	grid->year=grid->month=0; grid->time=grid->time_hyd=0;
	loct->C3ptn=1.0;loct->C4ptn=0.0;flux->Vanish(mass);
	(ecochar->c3)->grw_pd=(ecochar->c4)->grw_pd=0.0;
	(ecochar->c3)->gdd=(ecochar->c4)->gdd=0.0;
	for(int f=0;f<DAYNUM;f++){
		loct->dlen[f]=0.0;loct->sl_dec[f]=loct->sl_hgt[f]=0.0;
		loct->top_rad[f]=loct->gl_rad[f]=loct->gl_rad_g[f]=0.0;
		loct->rad_net_g[f]=loct->rad_net_p[f]=0.0;
		loct->par[f]=loct->apar[f]=loct->par_wm2[f]=loct->fapar[f]=0.0;
		loct->gd[f]=0;loct->gdd[f]=0.0;loct->prsr[f]=loct->dnsa[f]=loct->dnsa[f]=0.0;
		loct->vp[f]=loct->vps[f]=loct->vpd[f]=0.0;loct->slope_vps[f]=loct->r_aero[f]=0.0;
		loct->pm_evp[f]=loct->pm_trn[f]=loct->evpr[f]=loct->trspr[f]=0.0;
		loct->ro1[f]=loct->ro2[f]=loct->msnwa[f]=0.0;
		loct->msww[f]=loct->msw30[f]=loct->snp[f]=0.0;
		loct->snp[f]=loct->thaw[f]=0.0;loct->vmc30[f]=loct->vmc[f]=0.0;
		(ecochar->c3)->mgdd[f]=(ecochar->c4)->mgdd[f]=0.0;
		(ecochar->c3)->season[f]=(ecochar->c4)->season[f]=0;
		(ecochar->c3)->psat[f]=(ecochar->c4)->psat[f]=0.0;(ecochar->c3)->eK[f]=(ecochar->c4)->eK[f]=0.0;
		(ecochar->c3)->lue[f]=(ecochar->c4)->lue[f]=0.0;(ecochar->c3)->gs[f]=(ecochar->c4)->gs[f]=0.0;
		(ecochar->c3)->gc[f]=(ecochar->c4)->gc[f]=0.0;(ecochar->c3)->ci[f]=(ecochar->c4)->ci[f]=0.0;
		(ecochar->c3)->opt_lai[f]=(ecochar->c4)->opt_lai[f]=0.0;
		(ecochar->c3)->qTf[f]=(ecochar->c4)->qTf[f]=0.0;(ecochar->c3)->qTc[f]=(ecochar->c4)->qTc[f]=0.0;
		(ecochar->c3)->qTr[f]=(ecochar->c4)->qTr[f]=0.0;(ecochar->c3)->lf[f]=(ecochar->c4)->lf[f]=0.0;
		(ecochar->c3)->lc[f]=(ecochar->c4)->lc[f]=0.0;(ecochar->c3)->lr[f]=(ecochar->c4)->lr[f]=0.0;
		(ecochar->c3)->malloc_f[f]=(ecochar->c4)->malloc_f[f]=0.0;
		(ecochar->c3)->malloc_c[f]=(ecochar->c4)->malloc_c[f]=0.0;
		(ecochar->c3)->malloc_r[f]=(ecochar->c4)->malloc_r[f]=0.0;
```

```
                        (ecochar->soil)->albedo[f]=0.0;
                }
}
void Initiate::InitVegCond(Gchar *grid, Gcon *loct, Bmas *mass, Echar *ecochar)
{
                if(grid->whc<=0.0){grid->whc=374.524371;    /*      whc    whole    */}
                if(grid->wilt_point<=0.0){grid->wilt_point=145.039711; }
                /*    input    parameter */
                (ecochar->c3)->SetParameterC3(grid);(ecochar->c4)->SetParameterC4(grid);
                (ecochar->soil)->SetParameterSoil(grid);
                for(int f=0;f<12;f++){
                        grid->month=f;
                        (ecochar->c3)->GrowthPeriod(grid, loct);
                        (ecochar->c4)->GrowthPeriod(grid, loct);
                }
                /*    input initial    biomass*/
                InitC3(grid,mass);InitSoil(grid,mass);
                (mass->plant)->fol_p = (mass->c3)->fol_p + (mass->c4)->fol_p;
                (mass->plant)->stm_p = (mass->c3)->stm_p + (mass->c4)->stm_p;
                (mass->plant)->rot_p = (mass->c3)->rot_p + (mass->c4)->rot_p;
}
void Initiate::InitC3(Gchar *grid, Bmas *mass)
{
                double c3Init = 0;double c3InitRoot = 5.5;
                if(grid->vegNo == 2){
                        (mass->c3)->fol= c3Init; (mass->c3)->stm= c3Init; (mass->c3)->rot= c3InitRoot;
                        for(int i=0;i<DAYNUM;i++){
                                (mass->c3)->mfol[i]= c3Init; (mass->c3)->mstm[i]= c3Init;
                                (mass->c3)->mrot[i]= c3InitRoot;
                        }
                        (mass->c3)->fol_p= c3Init; (mass->c3)->stm_p= c3Init; (mass->c3)->rot_p= c3InitRoot;
                }else {
                        (mass->c3)->fol=INIT_MASS; (mass->c3)->stm=INIT_MASS; (mass->c3)->rot=INIT_MASS;
                        for(int i=0;i<DAYNUM;i++){
                                (mass->c3)->mfol[i]=INIT_MASS; (mass->c3)->mstm[i]=INIT_MASS;
                                (mass->c3)->mrot[i]=INIT_MASS;
                        }
                        (mass->c3)->fol_p=INIT_MASS; (mass->c3)->stm_p=INIT_MASS; (mass->c3)->rot_p=INIT_MASS;
                }
}
void Initiate::InitSoil(Gchar *grid, Bmas *mass)
{
                (mass->soil)->ltr=0;    (mass->soil)->msl=59;
                for(int i=0;i<DAYNUM;i++){
                        (mass->soil)->mltr[i]=0;    (mass->soil)->mmsl[i]=59;
                }
}
/*****    initialization    of    climatic    conditions    (Primary    data) ******/
void Initiate::InitClimateCond(Gchar *grid, Gcon *loct, Bmas *mass, Echar *ecochar)
{
                grid->tmp_sfc_am=0.0;  grid->tmp_sfc_mx=-100.0;grid->tmp_sfc_mn=100.0;
                grid->tmp_sfc_bio=0.0;  grid->xprate_sfc_ann=0.0;
                double n_bio=0.0;
                for(int h=0;h<DAYNUM;h++){
                        grid->month=h;
                        /*    CO2 condition */
                        grid->SetAtmCO2();
                        /*    climatic    conditions    */
                        grid->tmp_sfc[h]=grid->tmp_sfc_a[h]-ZAT;grid->tmp_2m[h]=grid->tmp_2m_a[h]-ZAT;
                        grid->tmp10_soil[h]=grid->tmp10_soil_a[h]-ZAT;grid->tmp200_soil[h]=grid->tmp200_soil_a[h]-ZAT;
                        grid->dswrf_toa[h]=grid->dswrf_toa_a[h];grid->dswrf_sfc[h]=grid->dswrf_sfc_a[h];
                        grid->tcdc_clm[h]=grid->tcdc_clm_a[h]/100.0;    grid->xprate_sfc[h]=dPrecip[h];
                        grid->spfh_2m[h]=grid->spfh_2m_a[h];grid->soilw10[h]=grid->soilw10_a[h];
                        grid->soilw200[h]=grid->soilw200_a[h];grid->wnd_10m[h]=grid->wnd_10m_a[h];
                        grid->whc30 = 0.1*grid->whc;
```

```
		/*	annual mean temperature	*/
		grid->tmp_sfc_am+=grid->tmp_sfc[h]*((double)grid->mm[h])/365.0;
		/*	annual	maximum	*/
		grid->tmp_sfc_mx=(grid->tmp_sfc[h]>grid->tmp_sfc_mx)?grid->tmp_sfc[h]:grid->tmp_sfc_mx;
		/*	annual	minimum	*/
		grid->tmp_sfc_mn=(grid->tmp_sfc[h]<grid->tmp_sfc_mn)?grid->tmp_sfc[h]:grid->tmp_sfc_mn;
		if(grid->tmp_sfc[h]>0.0){
			grid->tmp_sfc_bio+=grid->tmp_sfc[h];
			n_bio+=1.0;
		}
		/*	annual	total	precipitation	*/
		grid->xprate_sfc_ann+=grid->xprate_sfc[h];
		if(n_bio>1.0){grid->tmp_sfc_bio*=1.0/n_bio;
		}else{grid->tmp_sfc_bio=0.0;}
		/*	solar	decrination	and	solar	height	*/
		/*	radiation fluxes	and	day-length	*/
		loct->SetRadiation(grid, h);
	}
}
/*******	location	conditions	derived	from	the	primary	data (Secondary data1) *******/
void Initiate::InitLocationCond(Gchar *grid, Gcon *loct, Bmas *mass, Echar *ecochar,
		SimCalculation *cSCal, TransSimulation *pTrans)
{
	/***	water condition - Sim-HYDRO***/
	loct->sww= 0.025 * grid->whc;		//	whole	soil	water	content,	mm
	loct->sw30= 0.1 * grid->whc30;loct->snwa= 0.0;loct->dDeepWater = 50;
}
void Initiate::InitGridTrans(Gchar *grid, TransSimulation *pTrans)
{
	for(int e=3;e<9;e++){
		grid->tmp_sfc_a[e] = grid->tmp_sfc_a[e] +	pTrans->m_dTempRate;
		grid->tmp_2m_a[e] = grid->tmp_2m_a[e] +	pTrans->m_dTempRate;
		grid->tmp10_soil_a[e] = grid->tmp10_soil_a[e]	+	pTrans->m_dTempRate;
		grid->tmp200_soil_a[e] = grid->tmp200_soil_a[e] + pTrans->m_dTempRate;
		grid->tmp_sfc[e] = grid->tmp_sfc_a[e] - ZAT;
		grid->tmp_2m[e] = grid->tmp_2m_a[e]	- ZAT;
		grid->tmp10_soil[e] = grid->tmp10_soil_a[e] - ZAT;
		grid->tmp200_soil[e] = grid->tmp200_soil_a[e] - ZAT;
	}
	for( e=0;e<12;e++){grid->xprate_sfc[e] += grid->xprate_sfc[e] * pTrans->m_dPreRate ;}
}
```

```
// SimCalculation.h: interface for the Schar class.
#if !defined(AFX_SIMCALCULATION_H__64D2051E_670B_47E7_949D_CA5EA4CC437A__INCLUDED_)
#define AFX_SIMCALCULATION_H__64D2051E_670B_47E7_949D_CA5EA4CC437A__INCLUDED_
#if _MSC_VER > 1000
#pragma once
#endif
class SimCalculation
{
public:
		bool m_bTrace;
public:
		void CO2DynamicsStable(	Gchar *grid,	Gcon	*loct,
				Echar *ecochar,	Bmas	*mass,	Cflx	*flux, FILE *fp_r4, TransSimulation *pStable =NULL);
		void CO2DynamicsTrans(	Gchar *grid,	Gcon	*loct,
				Echar *ecochar,	Bmas	*mass,	Cflx	*flux);
		void WaterDynmcL( Gchar	*grid,	Gcon	*loct,	Bmas	*mass,	Echar *ecochar, int iFlag);
		SimCalculation();
		virtual ~SimCalculation();
};
#endif

// SimCalculation.h: interface for the Schar class.
#include "stdafx.h"
#include "SimCycleMFC_OP.h"
#include "SimCalculation.h"
#include<math.h>
#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif
SimCalculation::SimCalculation()
{		m_bTrace = false;}
SimCalculation::~SimCalculation()
{}
/********	dynamic	estimation	of	environmnetal	conditions	(Secondary data2) ********/
void SimCalculation::WaterDynmcL( Gchar	*grid,	Gcon	*loct,	Bmas	*mass,	Echar *ecochar, int iFlag)
{
		/**	air	conditions	**/
		loct->SetVariable(grid);
		/**	hydrological	water	budget	**/
		loct->WaterBudget(grid, ecochar, mass);
		/**	ecophysiology :	changing	**/
		(ecochar->c3)->EcophysiologyVeg(grid, loct, (mass->c3));
		(ecochar->c4)->EcophysiologyVeg(grid, loct,	(mass->c4));
}
void SimCalculation::CO2DynamicsStable(	Gchar *grid,	Gcon	*loct,
				Echar *ecochar,	Bmas	*mass,	Cflx	*flux, FILE *fp_r4, TransSimulation *pStable)
{
		long	f, term_time;double plantmass;double npp_t, nep_t, plant_t, soil_t;
		int iFlag=0; plantmass=0.0;npp_t=nep_t=plant_t=soil_t=0.0;
		for(f=0;f<DAYNUM;f++){
			grid->month=f;
			/*** environmental condition ***/
			WaterDynmcL(grid, loct,	mass, ecochar, iFlag);
			/*****	vegetation	processes	*****/
			flux->biome_processes(grid,loct,	ecochar, mass);
			flux->plant_stand(grid, loct,mass);
			/***** carbon supply from plant to soil, through litterfall *****/
			(flux->soil)->lL[f]=(flux->plant)->lL[f];
			/*****	soil	processes	*****/
			(flux->soil)->DoSoilProcess(grid,loct, (ecochar->soil), (mass->soil));
			/***** ecosystem	mass	balance *****/
```

```
		/*	net	ecosystem	production	*/
		flux->nep[f]=(flux->plant)->npp[f] - (flux->soil)->rS[f];
		/*	total	ecosystem	carbon	storage	*/
		mass->total[f] = (mass->c3)->plant[f]*loct->C3ptn + (mass->c4)->plant[f]*loct->C4ptn+
								(mass->soil)->soil[f];
		/** net carbon balance taking crop harvest into account **/
		flux->ncb[f]=flux->nep[f]+(flux->plant)->hvst[f];
		/*	annual	average	plant	mass	*/
		plantmass+=(mass->plant)->plant[f]*((double)(grid->mm[f]))/365.0;
		npp_t+=(flux->plant)->npp[f];
		nep_t+=flux->nep[f];
		plant_t+=(mass->plant)->plant[f]*((double)(grid->mm[f]))/365.0;
		soil_t+=(mass->soil)->soil[f]*((double)(grid->mm[f]))/365.0;
		/**	net	radiation	**/
		loct->F_net_rad(grid,	mass, ecochar);
		loct->ETBudget(grid, ecochar, mass);
		loct->SetAPAR(grid, mass, ecochar);
	}
	// Trace varible to stable stage.
	if(m_bTrace){
		if(grid->vegNo == 2){
			pStable->m_iRoundYear = 1;pStable->PrintData(grid, loct, ecochar, mass, flux);
		}
	}
}

void SimCalculation::CO2DynamicsTrans(	Gchar *grid,	Gcon	*loct,
		Echar *ecochar,	Bmas	*mass,	Cflx	*flux)
{
	long	f;double acnep,plantmass;double npp_t, nep_t, plant_t, soil_t;int iFlag=0;
	plantmass=acnep=0.0;npp_t=nep_t=plant_t=soil_t=0.0;
	for(f=0;f<DAYNUM;f++){
		grid->month=f;
		/*** environmental condition ***/
		WaterDynmcL(grid, loct,	mass, ecochar, iFlag);
		/*****	vegetation	processes	*****/
		flux->biome_processes(grid,loct,	ecochar, mass);
		flux->plant_stand(grid, loct,mass);
		/***** carbon supply from plant to soil, through litterfall *****/
		(flux->soil)->lL[f]=(flux->plant)->lL[f];
		/*****	soil	processes	*****/
		(flux->soil)->DoSoilProcess(grid,loct, (ecochar->soil), (mass->soil));
		/***** ecosystem	mass	balance *****/
		/*	net	ecosystem	production	*/
		flux->nep[f]=(flux->plant)->npp[f] - (flux->soil)->rS[f];
		/*	total	ecosystem	carbon	storage	*/
		mass->total[f] = (mass->c3)->plant[f]*loct->C3ptn + (mass->c4)->plant[f]*loct->C4ptn+
								(mass->soil)->soil[f];
		/** net carbon balance taking crop harvest into account **/
		flux->ncb[f]=flux->nep[f]+(flux->plant)->hvst[f];
		/*	annual	average	plant	mass	*/
		plantmass+=(mass->plant)->plant[f]*((double)(grid->mm[f]))/365.0;
		npp_t+=(flux->plant)->npp[f];
		nep_t+=flux->nep[f];
		plant_t+=(mass->plant)->plant[f]*((double)(grid->mm[f]))/365.0;
		soil_t+=(mass->soil)->soil[f]*((double)(grid->mm[f]))/365.0;
		acnep+=flux->ncb[f];	/*	annula	NCB, flux->nep[f];	*/
		/**	net	radiation	**/
		loct->F_net_rad(grid,	mass, ecochar);
		loct->ETBudget(grid, ecochar, mass);
		loct->SetAPAR(grid, mass, ecochar);
	}
}
```

```
// SimCycleMain.h: interface for the SimCycleMain class.
#if !defined(AFX_SIMCYCLEMAIN_H__073249D0_3437_4DFA_A815_AA1C2FD7A693__INCLUDED_)
#define AFX_SIMCYCLEMAIN_H__073249D0_3437_4DFA_A815_AA1C2FD7A693__INCLUDED_
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
#include "SimCalculation.h"
#include "PrintData.h"
class Initiate;
class SimCycleMain
{
public:
		int m_iTracer;FILE *fp_v[FILENUM];FILE *fp_s[12];FILE *fp_r3[FILENUM],*fp_r4[FILENUM];
		FILE *fp_ana[FILENUM];Gchar *grid;Gcon	*loct;Bmas	*mass;Cflx	*flux;Echar *ecochar;
		PrintData *m_cPrint;SimCalculation *m_cSimCal;Initiate *m_cInit;
public:
		// transitional function
		void OPTSimulation(TransSimulation *pTrans, TransSimulation *pStable);
		void OPTScheme(Gchar *grid, Gcon *loct, Echar *ecochar, Bmas *mass, Cflx *flux,
				FILE	*fp_v,	FILE	*fp_r3,	FILE	*fp_r4,	FILE	*fp_ana,	TransSimulation	*pTrans,	TransSimulation
*pStable);
		void MakeDir(CString DirName);
		void OPFileClose();
		void OPScheme(Gchar *grid, Gcon *loct, Echar *ecochar, Bmas *mass, Cflx *flux,
					FILE *fp_r3, FILE *fp_r4, FILE *fp_ana);
		void OPFileOpen();
		void OnePointSimulation(TransSimulation *pTrans = NULL);
		SimCycleMain();
		virtual ~SimCycleMain();
};
#endif

// SimCycleMain.cpp: implementation of the SimCycleMain class.
#include "stdafx.h"
#include "SimCycleMFC_OP.h"
#include "SimCycleMain.h"
#include "Initiate.h"
#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif
SimCycleMain::SimCycleMain()
{}
SimCycleMain::~SimCycleMain()
{}
void SimCycleMain::OnePointSimulation(TransSimulation *pTrans)
{
		OPFileOpen();//	open	source	files
		grid = new Gchar();	loct = new Gcon();	mass = new Bmas();
		flux = new Cflx(); ecochar = new Echar();
		m_cSimCal = new SimCalculation();
		m_cPrint = new PrintData();
		m_cInit = new Initiate();
		m_cInit->InitSim(grid);//	initialize	configuration
		// Location information of KBU site
		grid->raw= 516; grid->col= 3456;
		m_cInit->InitGrid(fp_s, grid);	// initialize grid condition
		grid->CO2y= g_iCO2YEAR; //Carbon dioxide
		int i=1; grid->area=i; grid->vegNo= i+1; //Vegetation number 2 : sparce grassland in GOGE
		OPScheme(grid, loct, ecochar, mass, flux, fp_r3[i], fp_r4[i], fp_ana[i]);
		OPFileClose();
		delete m_cInit;		delete m_cSimCal;	delete m_cPrint;
```

```
		delete flux;delete mass;delete loct;delete grid;
}
void SimCycleMain::OPScheme(Gchar *grid, Gcon *loct, Echar *ecochar, Bmas *mass, Cflx *flux,
								FILE *fp_r3, FILE *fp_r4, FILE *fp_ana)
{
		/** clear all parameters **/
		m_cInit->Clear(grid, loct, ecochar, mass, flux);
		/** initialize vegatation and soil conditions **/
		m_cInit->InitVegCond(grid, loct, mass, ecochar);
		/** initialize climate conditions **/
		m_cInit->InitClimateCond(grid, loct, mass, ecochar);
		/** initialize location conditions **/
		m_cInit->InitLocationCond(grid, loct, mass, ecochar, m_cSimCal);
		if(STATIC==1){
			/** carbon dynamics until stabilization **/
			m_cSimCal->CO2DynamicsStable(grid, loct, ecochar, mass, flux, fp_r4);
			/** outputs **/
			m_cPrint->PrintInFile(grid, loct, ecochar, mass, flux, fp_r3);
		}
}
void SimCycleMain::OPTSimulation(TransSimulation *pTrans, TransSimulation *pStable)
{
		OPFileOpen();// open source files
		grid = new Gchar();	loct = new Gcon();	mass = new Bmas();
		flux = new Cflx();	ecochar = new Echar();
		m_cSimCal = new SimCalculation();
		m_cPrint = new PrintData(1);
		m_cInit = new Initiate();
		m_cInit->InitSim(grid);// initialize configuration
		// Location information of KBU site
		grid->raw= 516; grid->col= 3456;
		m_cInit->InitGrid(fp_s, grid); // initialize grid condition
		grid->CO2y= g_iCO2YEAR;
		if((grid->lat<=NORTH&&grid->lat>=SOUTH)&&(grid->lon>=WEST&&grid->lon<=EAST)) grid->nnn++;
		int i=1; grid->area=i; grid->vegNo= i+1; //Vegetation number 2 : sparce grassland in GOGE
		OPTScheme(grid, loct, ecochar, mass, flux, fp_v[i], fp_r3[i], fp_r4[i], fp_ana[i], pTrans, pStable);
		OPFileClose();
		delete m_cInit;	delete m_cSimCal;	delete m_cPrint;
		delete flux;delete mass;delete loct;delete grid;
}
void SimCycleMain::OPTScheme(Gchar *grid, Gcon *loct, Echar *ecochar, Bmas *mass, Cflx *flux,
		FILE *fp_v, FILE *fp_r3, FILE *fp_r4, FILE *fp_ana, TransSimulation *pTrans, TransSimulation *pStable)
{
		/** clear all parameters **/
		m_cInit->Clear(grid, loct, ecochar, mass, flux);
		/** initialize vegatation and soil conditions **/
		m_cInit->InitVegCond(grid, loct, mass, ecochar);
		/** initialize climate conditions **/
		m_cInit->InitClimateCond(grid, loct, mass, ecochar);
		/** initialize location conditions **/
		m_cInit->InitLocationCond(grid, loct, mass, ecochar, m_cSimCal);
		/** carbon dynamics until stabilization **/
		m_cSimCal->CO2DynamicsStable(grid, loct, ecochar, mass, flux, fp_r4, pStable);
		/** outputs **/
		m_cPrint->PrintInFile(grid, loct, ecochar, mass, flux, fp_r3);
		// setting grazing routine,
		//if 1, then grazing ;if 0, then non-grazing
		grid->m_iSTFlag = 0;
		int iIncrease = 100;
		// transitional routine
		if(grid->vegNo == 2){
			for(int j = 1 ; j <= pTrans->m_iTSimYear ; j++){
				pTrans->m_iRoundYear = j;
				//m_cInit->InitLocationCond(grid, loct);
				if(j >= 1 && j <= iIncrease){
```

```
                                        grid->IncAtmCO2();
                                        m_cInit->InitGridTrans(grid, pTrans);
                                }
                                m_cSimCal->CO2DynamicsTrans(grid, loct, ecochar, mass, flux);
                                pTrans->PrintData(grid, loct, ecochar, mass, flux);
                        }
                }
}
void SimCycleMain::MakeDir(CString DirName)
{
                CFileFind finder;
                BOOL bExist = finder.FindFile(DirName);
                finder.Close();
                if(!bExist){
                        if ( !(CreateDirectory((LPCTSTR)DirName, NULL)))
                        {if (    GetLastError()    == ERROR_CANNOT_COPY ) return;}
                }
}
void SimCycleMain::OPFileOpen()
{
                int i;
                char cTemp[100];
                for(i =0 ; i < FILENUM ; i++){
                        strcpy(cTemp, GOGEDIR);fp_v[i]=fopen(strcat(cTemp,GOGENAME[i]),"rt");strcpy(cTemp, "");
                        strcpy(cTemp, R3DIR);      fp_r3[i]=fopen(strcat(cTemp,R3NAME[i]),"wt"); strcpy(cTemp, "");
                }
                for(i =0 ; i < 12 ; i++){
                        strcpy(cTemp, GOGEDIR);fp_s[i] =fopen(strcat(cTemp,CLIMATEDATA[i]),"rt");strcpy(cTemp, "");
                }
}
void SimCycleMain::OPFileClose()
{
                for(int i = 0 ;    i < FILENUM ; i++){fclose(fp_r3[i]);fclose(fp_v[i]);}
                for(i=0 ; i < 12; i++){     fclose(fp_s[i]);}
}
```