

Forcing PCR-GLOBWB with CRU data

Rens van Beek, Utrecht University, 2008

Introduction

The Climate Research Unit the University of East Anglia provides global climate data that potentially are extremely valid as forcing for macro-scale hydrological models as PCR-GLOBWB that requires precipitation, evapotranspiration and temperature. The advantage of the CRU products is that they are based on observations, covering the global land mass, and processed in a consistent manner. The period covered extends back over the past century, spanning the years 1901-2002 inclusive, for which sufficient meteorological data are available. As station density becomes low in certain regions of the world further back in time, a 30-year period was selected that was deemed to be little influenced by the alleged global warming at the end of the past century (1961-1990) which has since been adapted as the IPCC's baseline climate. Two products cover these respective periods with a spatial resolution of 0.5° and a monthly resolution, the first being the CRU TS 2.1 for the time series from 1901 to 2002 inclusive, the second being the CRU CLIM 1.0 for the climatology over 1961-1990. Both products contain primary variables -being the various temperature and precipitation products- that are directly obtained from station observations, which have a high density in space and time. Secondary variables with lower coverage are obtained by interpolation of the primary variables with the aid of empirical relationships. These secondary are particularly useful to calculate potential evapotranspiration that is required to force PCR-GLOBWB in absence of direct estimates of the actual evapotranspiration.

Some relevant fields, such as radiation and wind speed are only available for the climatology, which otherwise has been superseded by the time series itself and a new climatology product at a resolution of $10'$ (New et al., 1998; New et al., 1999; New et al., 2002; Mitchell and Hulme, 2005). Complications in the use of the CRU dataset arise from a varying extent of the landmass for some of the secondary variables, from a discrepancy with the land mask used by the hydrological model as well as from the coarse temporal resolution which may lead to undesired results when coupled with non-linear processes such as interception that operate on such finer time scales. Therefore, this report describes the downscaling of the CRU dataset to daily values with the assistance of the ERA-40 reanalysis, its extrapolation to the desired land mask of PCR-GLOBWB and to develop the methodology to calculate potential evapotranspiration for the different land surfaces in PCR-GLOBWB. The objectives of this exercise were to:

- Construct patches to extrapolate CRU data to a common land mask for all products and to extrapolate all relevant products to the land mask employed by PCR-GLOBWB;
- Calculate the monthly reference potential evaporation from the CRU dataset for the length of the time series;
- To develop a climatology of crop factors for the different land surfaces in PCR-GLOBWB that can be used to convert the monthly reference potential evapotranspiration into vegetation specific values;
- Break-down monthly values of CRU precipitation and temperature and the calculated reference potential evapotranspiration to daily values, gridded at 0.5° using daily surface fields from the ERA-40 reanalysis to drive the model.

In the next sections the methodology and resulting parameterizations will be discussed. This report does not investigate the quality issues related to the use of the CRU datasets for continental runoff.

Methodology

Processing the CRU datasets

The CRU datasets contain an ASCII file with the gridded integer surface fields for all months and a header with meta-information on the name, unit and conversion factor from integers to real values for each variable. The CRU TS 2.1 contains as primary variables the monthly precipitation total, the mean daily temperature and the diurnal temperature range per month. As secondary data, also the daily minimum and maximum temperature are provided.

Additional secondary datasets comprise the fractional cloud cover, the vapour pressure and the number of wet and frost days. In addition to these data, the CRU CLIM 1.0 provides secondary data on wind speed and radiation. Otherwise, this dataset is obsolete.

Each of the ASCII files with gridded fields was processed by means of a Fortran program (`cru_grid2asc.exe`, see Appendix 1) and stored as an individual ASCII grid with integer values (360 rows x 720 columns). This grid was converted to the PCRaster format by invoking the program `asc2map` from the python script `mapgen_cru.py`. This resulted in 1224 monthly gridded surface fields over the period 1901-2002 for the time series and 12 mean monthly maps for the climatology.

Specific processing of these surface fields was carried out by means of the python script `cru_proc.py` (Appendix 2). First, the PCRaster maps were extracted from the respective zip archives and their mean and standard deviation computed for the baseline period 1961-1990. This concerned the mean daily temperature, precipitation, cloud cover, vapour pressure, number of wet days and diurnal temperature range that were required for later processing. The CRU TS 2.1 dataset employs a common land mask for all its products. It misses wind speed, necessary to calculate the reference potential evapotranspiration but this is provided by the CLIM 1.0 dataset, albeit with a slightly different land mask. To extrapolate these data to the common land mask, a patch was constructed by determining the Holdridge life zone for each cell from the annual total precipitation and average temperature as approximation of the biotemperature. The potential evapotranspiration ratio (PET) was neglected as it originally is an empirical function of the biotemperature and does not add directly to the classification (Leemans, 1990). Temperature from the climatology over 1961-1990 was broken down logarithmically in the classes $< 1.5\text{ }^{\circ}\text{C}$, $< 3\text{ }^{\circ}\text{C}$, $< 6\text{ }^{\circ}\text{C}$, $< 12\text{ }^{\circ}\text{C}$, $< 16\text{ }^{\circ}\text{C}$, $< 24\text{ }^{\circ}\text{C}$ and $\geq 24\text{ }^{\circ}\text{C}$. Each of these classes was then broken down on the basis of precipitation, again logarithmically from $< 125\text{ mm/year}$ to more than 8000 mm/year , giving a total of 56 possible classes. For each of these classes, any cell of the CRU TS 2.1 product that was not covered by the CLIM 1.0 was assigned the ID of the nearest cell of the latter product belonging to the same Holdridge life zone. Thus, cell values could be extrapolated to the common land mask and used accordingly.

Reference potential evapotranspiration and crop factors

According to the FAO Guidelines (Doorenbos & Pruitt, 1977; Allen et al., 1998), potential evapotranspiration was calculated for a reference surface, ET_0 [$\text{L}\cdot\text{T}^{-1}$], named reference potential evapotranspiration hereafter, and converted to a crop specific potential evapotranspiration, ET_c [$\text{L}\cdot\text{T}^{-1}$], by means of a crop factor, k_c [-]:

$$ET_c = k_c \cdot ET_0. \quad \text{Equation 1}$$

In this manner, all meteorological influences are captured by the reference potential evapotranspiration whereas the crop factor, k_c , captures the effect of the individual plants and surface conditions on both the crop transpiration and the soil evaporation alike. This crop factor applies to a disease-free crop under an optimum supply of water and nutrients but local and regional environmental factors can be considered (Doorenbos & Pruitt, 1977). Although the crop factor approach was originally developed for real crops, it can be expanded to natural vegetation. Here, crop factors are thought to represent average conditions over a uniform, vegetated surface for a fixed period of time and computed and aggregated accordingly as explained below.

According to Allen et al. (1998) a new definition of the reference surface and the method to obtain the reference evapotranspiration were required as a result of the ambiguities that had arisen since the formulation of these concepts by Doorenbos & Pruitt in the FAO Guidelines of 1977. Thus, Allen et al. (1998) recommended replacing the originally preferred Penman Equation (Penman, 1948) by the Penman-Montheith Equation (Montheith, 1965) to calculate the reference potential evaporation. Through the inclusion of the canopy resistance, the Penman-Montheith Equation performs relatively accurate and consistent in both arid and humid climates whereas the Penman Equation was shown to overestimate the evapotranspiration frequently. Here the following version of the Penman-Montheith Equation has been used rather than the computational formula presented by Allen et al. (1998).

$$ET_0 = \frac{\delta(R_n - G) + \rho_a c_p \frac{(e_s - e_a)}{r_a}}{\lambda_v \left(\delta + \gamma \left(1 + \frac{r_s}{r_a} \right) \right)}, \quad \text{Equation 2}$$

where e_s is the saturation vapour pressure, e_a is the actual vapour pressure, both in [Pa], δ is the slope of the function of the saturation vapour pressure versus the air temperature [$\text{Pa} \cdot \text{C}^{-1}$], γ is the psychrometric constant [$\text{Pa} \cdot \text{C}^{-1}$], ρ_a is the density of air [$1.205 \text{ kg} \cdot \text{m}^{-3}$], c_p is the specific heat capacity of air [$1004 \text{ J} \cdot \text{kg}^{-1} \cdot \text{C}^{-1}$], R_n is the net incoming radiation and G the ground flux, both in [$\text{W} \cdot \text{m}^{-2}$], λ_v is the latent heat of vaporization [$\text{J} \cdot \text{kg}^{-1}$], and r_s and r_a are the surface and aerodynamic resistance respectively [$\text{s} \cdot \text{m}^{-1}$].

Following the FAO guidelines on crop water demands (Doorenbos & Pruitt, 1977), the reference potential evapotranspiration was calculated for longer periods, in this case the monthly resolution of the CRU dataset. On this ground, the ground flux was neglected, assuming no heat exchange between soil and air over longer periods. Of the variables of Equation 2, only ρ_a and c_p are constants and the vapour pressure is specified directly by the CRU dataset. All other variables had to be calculated. So, the saturated vapour pressure was calculated from the average daily temperature under the underlying assumption of the isothermal conditions of the Penman Equation (Allen et al., 1998):

$$e_s = 611 \cdot \exp\left(\frac{17.27 + \bar{T}}{\bar{T} + 237.3}\right). \quad \text{Equation 3}$$

\bar{T} denotes here the average monthly temperature from the CRU dataset in [$^{\circ}\text{C}$], which is referred to as T hereafter.

The slope of the saturation vapour pressure versus the air temperature is calculated by (Allen et al., 1998):

$$\delta = \frac{4098 \cdot e_s}{(T + 237.3)^2}. \quad \text{Equation 4}$$

The psychrometric constant, γ , is calculated as (Allen et al., 1998):

$$\gamma = \frac{c_p \cdot P}{\epsilon \lambda_v}, \quad \text{Equation 5}$$

where P is the atmospheric pressure [Pa], e is the ratio of the molecular weight of water vapour and dry air [0.622 -].

The psychrometric constant is not constant as both the latent heat of vaporization and the atmospheric pressure vary with the air temperature and the elevation respectively. Allen et al. (1998) ignore the variation in λ_v with T but this relationship is maintained here (Dingman, 1994):

$$\lambda_v = 2.501 \cdot 10^6 - 2370T. \quad \text{Equation 6}$$

Due to this temperature-dependency of λ_v the calculated evapotranspiration of Equation 6 is slightly larger at higher temperatures and vice versa than with the computational formula of Allen et al. (1998).

The atmospheric pressure is calculated here from a generalization of the gas law, assuming an overall temperature of 20 °C, assuming a typical atmospheric pressure at sea level of 101300 Pa (Allen et al., 1998):

$$P = 101300 \left(\frac{293 - 0.0065z}{293} \right)^{5.26}, \quad \text{Equation 7}$$

where z is the elevation in [m], which in this case was given by the average elevation of the Hydro1k land mass within each cell.

The effectiveness by which heat and vapour can be exchanged with the atmosphere depend on the vertical transport capacity of turbulent air, as expressed by the friction velocity of the eddies. Allen et al. (1998) estimate the diffusivity as the product of the friction velocity for the momentum transfer and the heat and vapour transfer:

$$r_a = \frac{\ln\left(\frac{z_m - z_D}{z_{0m}}\right) \ln\left(\frac{z_h - z_D}{z_{0h}}\right)}{k^2 \bar{u}_z}, \quad \text{Equation 8}$$

where m and h stand for the momentum and heat and vapour transfer respectively. z_m and z_h are respectively the heights at which the wind speed and temperature/vapour pressure are measured, z_D is the zero plane displacement height, z_0 stands for the roughness length, k is the Karman constant, [0.41 -], and \bar{u}_z is the monthly wind speed measured at height z_m .

In absence of wind speed fields in the CRU TS 2.1, the monthly climatology from CRU CLIM 1.0 was used to estimate \bar{u}_z . This value was capped at its lower end by 0.1 m·s⁻¹ as some negative values were present over certain parts of the world, most noticeably over the mountains in SE Australia. The wind speed was taken to be representative for that at 10 m,

the air temperature, from which e_s was calculated, at 2 m. The zero plane displacement was set at 2/3 of the vegetation height. The roughness length, Z_{0m} , for momentum transfer at 0.123 times the vegetation height whilst that for heat and vapour transfer, Z_{0h} , was set to 0.1 times Z_{0m} .

For each month, the incoming net radiation, R_n [$W \cdot m^{-2}$], was estimated as:

$$R_n = (1 - \alpha)R_s + R_l \quad \text{Equation 9}$$

where R_s is the incoming global shortwave radiation, R_l is the emitted long-wave radiation emitted by the earth surface, negative in general, and α is the albedo [-] determining the amount of shortwave radiation that is reflected back into the atmosphere.

R_s was obtained from the extraterrestrial radiation, R_a . The extraterrestrial radiation was calculated for a standard year as a function of the Julian day number and latitude and averaged per month (Dingman, 1994), giving 12 maps. Thus, the extraterrestrial radiation replicates the typical seasonal change over the globe with some zonal variations arising due to the interaction of day length and the angle of the incoming solar radiation. R_s was then calculated from the fraction of actual sunshine hours over the maximum sunshine hours as originally proposed by Doorenbos & Pruitt (1977):

$$R_s = \left(a + b \frac{n}{N} \right) R_a, \quad \text{Equation 10}$$

where n/N is the fraction of actual sunshine hours over the maximum sunshine hours and a and b are empirical parameters that are respectively set to 0.25 and 0.50 (Doorenbos & Pruitt, 1977).

Rather than fractional sunshine hours n/N the CRU CLIM 1.0 and TS 2.1 specify fractional cloud cover and a relation between the two variables was required. To this end, the tabulated data of Doorenbos & Pruitt (1977) were used, which also were applied by New et al. (1998) in the construction of the CRU datasets to standardize the available observations of fractional sunshine hours and cloud cover prior to interpolation. New et al. (1998) observed that bias was introduced at high latitudes due to weak sunshine at low sun angles in winter whilst in tropical regions cloud cover is often overestimated by observers. Given 1088 stations with joint observations of n/N and cloud cover, they applied additional empirical adjustments to reduce these errors. Hence, the reported, corrected cloud cover has been translated into fractional sunshine hours from the tabulated data of Doorenbos & Pruitt (1977) by means of linear interpolation (Figure 1).

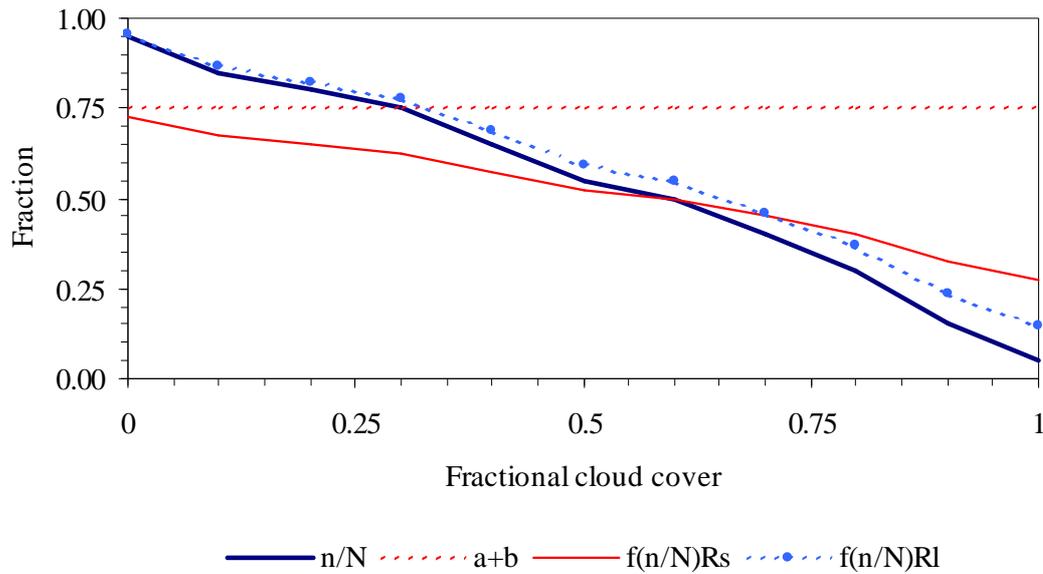


Figure 1: Relation between fractional cloud cover and fractional sunshine hours. $a+b$ are the empirical constants of Equation [10], determining the maximum incoming shortwave radiation for a cloudless sky, $f(n/N)R_s$ is the fraction according to Equation 9 of extraterrestrial radiation reaching the surface and $f(n/N)R_l$ is the reduction factor of the emitted long-wave radiation with increasing cloud cover.

Net long-wave radiation was calculated from the Stefan-Boltzman Equation for both the incoming and outgoing long-wave radiation given the respective emissivity of the atmosphere and surface and the body temperature. For most surfaces, the emissivity can be taken as one. For clear skies, the atmospheric emissivity depends largely on humidity, whilst clouds act as blackbodies and increase the emissivity of the atmosphere. These effects on the net emitted long-wave radiation are captured by the formula proposed by Allen et al. (1998) through the vapour pressure and the relative incoming global radiation, assuming that under isothermal conditions both the earth surface and the overlying air are at the same temperature:

$$R_l = -\sigma T_K^4 \left(0.34 - 4.43 \cdot 10^{-3} \sqrt{e_a} \right) \left(1.35 \frac{R_s}{R_0} - 0.35 \right), \quad \text{Equation 11}$$

where σ is the Stefan-Boltzman constant [$5.68 \cdot 10^{-8} \text{ W} \cdot \text{m}^{-2} \cdot \text{K}^{-4}$], T_K is the air temperature in Kelvin and R_0 is the incoming global radiation under clear-sky conditions, $(a+b)R_a$ (see Figure 1).

The reduction due to humidity ranges between 0.34 for completely dry air to zero when the vapour pressure exceeds 5.9 kPa. The correction due to the relative incoming global radiation ranges between 1 under clear-sky conditions to 0.1 under completely clouded conditions (see also Figure 1).

Not specified so far are those variables that are dependent on the surface conditions being the albedo, α , the surface resistance, r_s , and the vegetation height, z_{veg} . As a reference surface Allen et al. (1998) proposed hypothetical grass cover on the ground that this crop is well studied. This reference surface is defined unambiguously as “a hypothetical reference crop with an assumed crop height of 0.12 m, a fixed surface resistance of 70 s m^{-1} and an albedo of 0.23.” These values have been included in the PCRaster script to calculate reference potential evapotranspiration, `penmon_cru.txt` (see Appendix 3).

To convert the reference potential evapotranspiration to crop-specific values, crop factors are required (Equation 1). Conform the land surface parameterization in PCR-GLOBWB, these crop factors have to be specified for the fraction open freshwater, short vegetation and tall vegetation respectively and effective values have been calculated for each 0.5° cell. These crop factors compensate for any effects not taken into consideration in the calculation of the reference potential evapotranspiration. For water, this may concern the delayed exchange of heat between the air and deeper water bodies, lower surface roughness and lower albedo. For vegetated surfaces, it concerns amongst others the stomatal resistance and active leaf area as well as vegetated area and surface roughness. To account for temporal variations in these surface conditions, monthly crop factors were calculated according to the FAO Guidelines of Allen et al. (1998). These crop factors were based on the climatology over the period 1961-1990 as calculated from the CRU TS 2.1 dataset.

For the vegetated surface, temporal variations in the crop factor were assumed to correspond with the phenology of the vegetation over the growing season, prescribing the different stages of crop or plant development and vigour. These plant development stages can be subdivided into i) development proper, ii) full-cover or mid-season stage, iii) senescence, and iv) dormancy (Figure 2.a). The development stage is the moment between emergence of the first leaves or plants –depending whether the vegetation is perennial or annual- and the attainment of full cover. During the mid-season period vegetation is at its most vigorous, covering the largest area with the most active and healthy leaves partaking in transpiration. After full cover has been achieved, senescence set in, with leaves becoming less active and healthy, *e.g.*, in the fall for temperate regions, until dormancy sets in. These stages pertain to natural vegetation that will exploit the available growing conditions fully. For actual crops, the length of these stages may be different, with full growth periods ranging between 100 to 150 days on average depending on the favourability of the meteorological conditions during the growing season (Doorenbos & Pruitt, 1977). The length of the full growing season is determined by the seasonal changes in radiation, temperature and moisture availability, as reproduced by the CRU TS 2.1 climatology and assuming that all other factors, *e.g.*, fertility, are optimal. Here, temperature is used as a proxy for the limitations temperature and radiation pose to photosynthesis. Generally, growth is deemed impossible whenever killing frost occurs or the air temperature is below 5°C for long periods. This threshold is applied here to climatology of mean monthly temperature, which statistic captures the likelihood of killing frost in the long run. At higher latitudes, this temperature threshold also reflects the lack of radiation during winter. The seasonal course in temperature-driven growth was simulated by the BATS scheme (Dickinson et al., 1983):

$$f(T) = 1 - \left(\frac{T_{max} - T}{T_{max} - T_{min}} \right)^2 \quad \text{Equation 12}$$

where T is the monthly temperature, $T_{min} = 5^\circ\text{C}$ and T_{max} is the maximum temperature over the climatology, subject to the condition $T \geq 5^\circ\text{C}$. $f(T)$ is the temperature function that varies between 0 and 1 for $T = T_{min}$ and $T = T_{max}$ respectively.

T_{min} was kept at 5°C to reflect potentially favourable growth conditions in warmer climes. T_{max} was allowed to vary as locally vegetation will exploit the growing conditions fully.

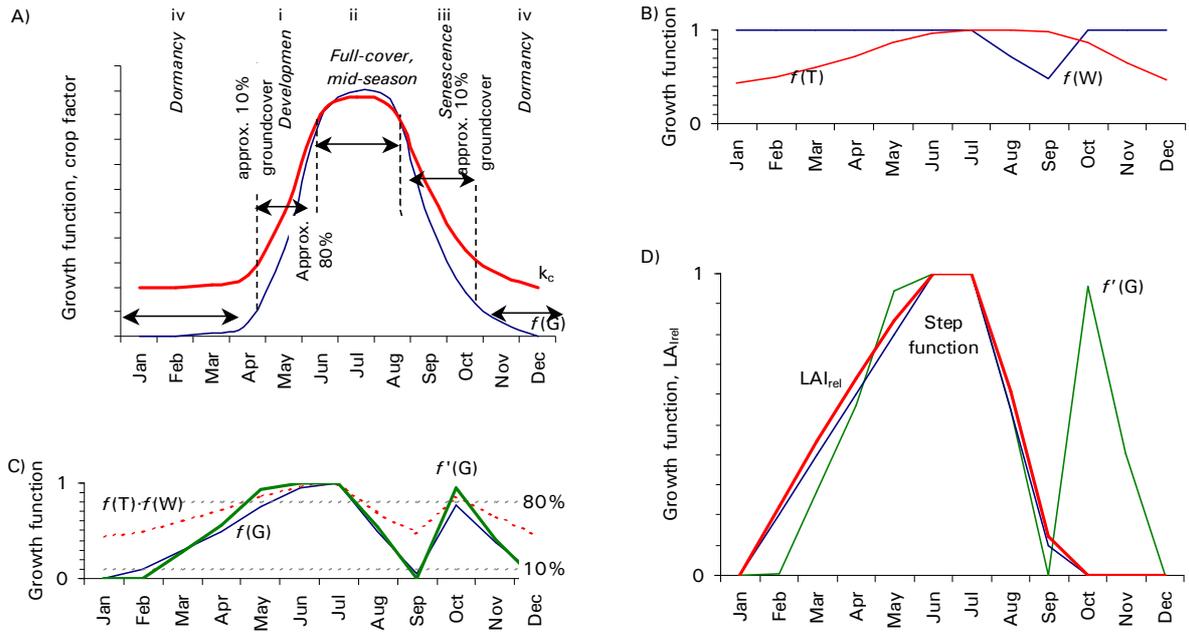


Figure 2: Delineation of the growing season

Delineation of the growing season: a) Growth stages adapted after Doorenbos and Pruitt (1977) for the growth function $f(G)$ with corresponding crop factor; b) dependence of the growth function on temperature, $f(T)$, and the cumulative moisture deficit, $f(S)$; c) derivation of the growth function: raw growth function $f(T) \cdot f(W)$, initial growth function scaled between 0 and 1, $f(G)$, and $f'(G)$ subdivided into growth stages on the basis of thresholds for emergence and full-cover (10 and 80% respectively); d) $f'(G)$, derived step function and relative LAI, LAI_{rel} , after modification of the step function with the exponent a .

Shown is the curve for S Spain (36.75°N, 4.25°W) with severe water shortage over the summer. The tuned maximum crop factor for this cell amounts to 0.56.

Water stress will reduce the potentially growth and this was taken into account as a function of the cumulative moisture deficit over the year:

$$f(W) = \max\left(0, 1 + \frac{\sum \Delta W}{W_{req}}\right), \quad \text{Equation 13}$$

where W is the available moisture storage [mm], $\sum \Delta W$ is the cumulative moisture deficit over the months that the moisture storage is exhausted and W_{req} the required moisture to sustain vigorous growth. $f(W)$ is the moisture stress function that varies between 1 and 0 for conditions of no stress ($\sum \Delta W = 0$) to severe stress ($\sum \Delta W \geq W_{req}$).

The water balance equation underlying Equation 13 was applied to the climatology of the CRU TS 2.1 dataset until a dynamic equilibrium was obtained:

$$W_t = W_{t-1} + P_t - k_c \cdot E_{0t} \quad \text{Equation 14}$$

where t and $t-1$ are used to denote the current and previous month respectively, W is the moisture storage [mm], P is the monthly precipitation total and E_0 is the reference potential evapotranspiration, both in [mm], and k_c is a tentative crop factor for the vegetation [-].

Whenever W_t is below zero, it is added to the cumulative moisture deficit, if it is above zero, the cumulative moisture deficit is reset to zero. If the moisture storage exceeds the total available moisture storage capacity it is set to that value and the excess considered to be removed by drainage or runoff. This maximum soil moisture storage capacity was set to 300 mm conform the Thornthwaite model for soil moisture availability. The tentative crop factor

varied between the crop factor for bare soil, k_s , and that of the vegetation at full-cover, k_{full} , with the temperature growth function. The full-cover crop factor was decreased from 1.2 to 0.2, the range of k_{full} to k_s according to Allen et al. (1998) by 0.01 increments until the following conditions were met: i) the maximum available storage should exceed W_{req} , the moisture storage required for vigorous growth, and ii) the maximum accumulated moisture deficit should not fall below $-W_{req}$. The adaptation of this crop factor is thought to imitate to the adaptations by the vegetation to prolonged conditions of water stress through vegetation density and stomatal control. In this case W_{req} was arbitrarily set to 125 mm, the upper limit of the class of deserts in the Holdridge life zone classification. When the adaptation of the tentative crop factor did not result in any positive soil moisture storage over the year, growth was deemed to be entirely driven by erratic rainfall. In these cases, $f(W)$ was calculated alternatively as:

$$f(W) = \frac{P}{P_{max}}, \quad \text{Equation 15}$$

where P is the monthly precipitation total and P_{max} the maximum over the climatology [mm].

The growth function, $f(G)$, was then obtained as the product of $f(T)$ and $f(W)$ and scaled by its minimum and maximum to range between 0 and 1 (Figure 2). For its rising limb, this growth curve represents the development of ground cover over time, the maximum coinciding with the moment of most vigorous growth, its minimum with dormancy. This growth curve was subdivided into the four development stages on the following criteria conform Doorenbos and Pruitt (1977):

- Vegetation emerges after dormancy as soon as $f(G)$ exceeds 0.10 [-];
- Vegetation becomes fully dormant once $f(G)$ falls below 0.10 [-];
- The full growing season extends from emergence to the onset of dormancy, including the optimum growth conditions;
- Full-cover during the mid-season stage is attained in the period that the growth conditions is optimal and $f(G)$ exceeds 0.80 [-];
- Senescence of the vegetation sets in when the growth curve starts to decline and $f(G)$ falls below 0.60 [-].

On the basis of these criteria, the different points (emergence, full-cover, and senescence) were connected by straight lines whilst from the onset of senescence the falling limb was extrapolated to create a step function to relate the growth function to the relative change in the amount of active leaves participating in transpiration ($f'(G)$ scaled to range between 0 and 1). This fraction is taken to be proportional to the green LAI or *leaf area index* that is related directly to the crop factors by Allen et al. (1998). For vigorous vegetation, the increase in LAI will be fast during the early stages of growth and senescence and slower during the later stages. This is achieved by applying an exponent to the step function so:

$$LAI_{rel} \propto f'(G)^a. \quad \text{Equation 16}$$

This exponent a varies with the occurrence of prolonged water stress and is parameterized as:

$$a = 1 - 0.5 \frac{k_{full} - k_s}{k_{full_{max}} - k_s}, \quad \text{Equation 17}$$

where k_{full} is the tuned, tentative crop factor from the water balance calculation and $k_{full_{max}}$ and k_s are the maxima for full-cover and bare soil (1.2 and 0.2 [-] respectively).

The exponent a ranges from 0.5 when k_{full} equals $k_{full_{max}}$ and 1.0 when k_{full} equals k_s . These limits were taken from Allen et al. (1998) and reflects that LAI does not fall in direct

proportion to stand density under vigorous growth conditions. This exponent is modified when the relative LAI function deviates from $f'(G)$:

$$a = a + 0.5(LAI_{rel} - f'(G)). \quad \text{Equation 18}$$

In this manner, the relative LAI increases when the growth function lies above the step function and decreases when it lies below it (Figure 2).

Allen et al. (1998) related the crop factor to the LAI in the following way:

$$k_c = k_{c_{min}} + (k_{c_{full}} - k_{c_{min}}) \left(1 - \exp[-0.7LAI]\right), \quad \text{Equation 19}$$

where $k_{c_{min}}$ is the minimum crop factor for bare soil [$\approx 0.15-0.20$ -] and $k_{c_{full}}$ is the estimated crop factor under full-cover conditions.

Allen et al. (1998) recommended Equation 19 to estimate the crop factor during the mid-season stage for annual types of vegetation that are either natural or in a non-pristine state, *i.e.*, stands that do not attain optimum stand conditions in terms of height and density. Here, this equation is applied globally to all vegetation types and to the course of the LAI over the year given that the relative change in the LAI includes the wetting period during the initial stage for crops and natural vegetation alike and that vegetation will exploit the available growing season fully, either through competition or human intervention (multiple cropping systems, forage and green fertilization, *etc.*).

The actual values of the LAI were based on the GLCC version 2 global land cover dataset with a resolution of 30 arc seconds (30" or nominally 1 km). The Olson ecosystem classification (Olson, 1994a, b) was preferred as this provides the most detailed distinction between natural vegetation and crop types in the different climate zones (Appendix 4). Moreover, estimates of the LAI during the growing season and dormancy for these classes could be obtained from Hagemann et al. (1999) so that the monthly LAI, from which the monthly crop factor was to be estimated, could be obtained from:

$$LAI = LAI_d + LAI_{rel} (LAI_g - LAI_d), \quad \text{Equation 20}$$

where LAI_{rel} is the relative change in the LAI over the year [0-1 -] and LAI_d and LAI_g are the LAI [$m^2 \cdot m^{-2}$] for each vegetation class respectively.

For each class, these LAI values represent the effect of the seasonal and local conditions, *e.g.*, water stress, dormancy, fertility and human intervention, on plant type and density. These seasonal LAI values range between 0 and $9.9 m^2 \cdot m^{-2}$. No seasonality is present for some tropical vegetation types, deserts as well as for built-up areas (10 out of 73 classes excluding open water). Whenever the LAI is zero (12 out of 73 classes for LAI_d , 5 for LAI_g), the estimated crop factor will revert to that for bare soil evaporation, which is set to the upper limit of 0.20 [-] reported by Allen et al. (1998). Also, the estimated crop factor for the mid-season stage will be less than 95% of the estimated value for full cover conditions, $k_{c_{full}}$, whenever LAI_g is less than $4.3 m^2 \cdot m^{-2}$ (35 classes for LAI_g).

The estimate for the mid-season crop factor under full-cover conditions under sub-humid, calm wind conditions –minimum daily relative humidity of 45% and a wind speed at $2 m \cdot s^{-1}$ at 2 m height- was obtained by (Allen et al., 1998):

$$k_{c_{full}} = 1.0 + 0.1h, \quad \text{Equation 21}$$

where h is the height of vegetation [m] and $k_{c_{full}}$ is limited to 1.2 m when $h > 2$ m.

This value is corrected for meteorological conditions other than sub-humid or calm wind speeds during the mid-season stage as follows (Allen et al., 1998):

$$k_{c_full} = k_{c_full} + [0.04(u_2 - 2) - 0.004(RH_{min} - 45)] \left(\frac{h}{3}\right)^{0.3}, \quad \text{Equation 22}$$

where u_2 is the wind speed at 2 m [$\text{m}\cdot\text{s}^{-1}$], RH_{min} is the minimum daily relative humidity [%] and h is the vegetation height.

The meteorological correction entails an increase in the crop factor when the mid-season wind speed is higher than $2 \text{ m}\cdot\text{s}^{-1}$ and the minimum daily relative humidity is less than 45%. and *vice versa*. The wind speed was obtained from the CRU CLIM 1.0 climatology and multiplied by 0.75 [-] to scale it from 10 m to 2 m height and limited between 1 and $6 \text{ m}\cdot\text{s}^{-1}$ (*cf.* Allen et al., 1998). The minimum relative humidity was estimated as 100% x the ratio of the saturated vapour pressure for the minimum and maximum daily temperature, which could be estimated from the CRU climatology by adding or subtracting half the daily temperature range from the mean monthly temperature. Here, the minimum temperature is used as an approximation of the dew point temperature and therefore reduced by 2°C in arid and semi-arid regions, i.e., when cumulative water stress is present. RH_{min} is limited between 20 and 80% (*cf.* Allen et al., 1998). For both u_2 and RH_{min} the mean monthly values over the mid-season stage were used.

The vegetation height is limited between 0.1 and 10 m (Allen et al., 1998). Subject to these constraints, the vegetation height itself was estimated from the dataset of Hagemann et al. (1999) by dividing the vegetation roughness length, z_{0_veg} [m], by 0.123 [-]. This returns the maximum of 10 m for most forest types (6 classes out of 73) and the minimum of 0.1 for most barren surfaces as deserts and glaciers (7 classes).

In order to calculate the monthly crop factors at the spatial resolution of 0.5° , the phenology and the relative LAI, as well as the meteorological part of the correction factor of Equation 22, were derived at 0.5° by means of a PCRaster Python script (see Appendix 2). The vegetation-dependent factors were processed by means of an AML script in ArcInfo and the 0.5° monthly maps resampled to the GLCC resolution of $30''$. This results into monthly grids at $30''$ that specify the crop factor for the Olson class for each cell. These grids have been aggregated to a resolution of 0.5° cells and converted back to PCRaster maps as input to PCR-GLOBWB. As the scaling of the reference potential evapotranspiration to the crop-specific potential evapotranspiration is linear (Equation 1), the aggregated value is the mean of all $30''$ cells. To link the crop factors to the surface types of short and tall vegetation used in PCR-GLOBWB, the subdivision of Appendix 4 has been used, with an additional distinction being made between natural vegetation, rain fed crops and irrigated crops in order to substitute any of these categories by other estimates if desirable. Hence, the overall crop factor for each vegetation type was calculated as the weighed average of each category per surface type, given their fractional cover within each grid cell.

Over open water the potential evaporation differs from the reference potential evapotranspiration as a result of the difference in albedo, surface roughness and the heat storing capacity of deep water that influences heat transfer. Hence, a cell identified as a lake or reservoir was characterized as deep water, whereas the remaining cells with rivers were characterized as shallow water. For the river cells, the crop factor for open water applied to the channel area itself, as prescribed by the channel width and length used in PCR-GLOBWB, whilst an additional crop factor applied to the remaining wetland area. Thus, the monthly crop factor for the fraction open fresh water in each cell was computed as:

$$k_{c_water} = k_{c_deep}, \quad \text{if identified as lake or reservoir, else}$$

$$k_{c\ water} = \frac{\Delta X_{channel} \Delta L_{channel} k_{c\ shallow} - (f_{water} \Delta X^2 - \Delta X_{channel} \Delta L_{channel}) k_{c\ wetland}}{f_{water} \Delta X^2}, \text{ Equation 23}$$

where k_c is the crop factor [-] for the entire fresh water surface, deep and shallow water and wetland area respectively, $\Delta X_{channel}$ and $\Delta L_{channel}$ are the channel width and length [m], f_{water} is the fraction of land area within each cell [-], and ΔX_2 is the land surface area within each cell [m²].

For all freshwater surfaces it was assumed that evapotranspiration could only occur over ice-free surfaces. This was achieved by excluding all months for which the mean monthly air temperature was below 0°C.

Ice-free shallow water was assumed to have no appreciable heat storing capacity and its crop factor assumed to be constant with time. It was calculated according to Equation 21 with a vegetation height of 0.1, the minimum height according to Allen et al. (1998), to account to the varying influence of surrounding vegetation on the roughness as many rivers have widths much smaller than the nominal 1 km GLCC resolution. The same value was applied to lakes and reservoirs in the tropics, which were taken equivalent with an average annual air temperature of 20°C. Elsewhere, deep water was thought to have some potential heat storing capacity and wetland vegetation to develop over the growing season, provided that the surface was ice-free. In both cases, this was simulated by the BATS scheme of Equation 12 using the mean monthly air temperature from the 1961-1990 climatology with T_{min} and T_{max} being defined by the minimum and maximum air temperature over the period that $T > 0^\circ\text{C}$.

Outside the tropical region, the crop factor may decrease by 20 to 30% in early spring due to heat storage and increase by a similar value in late autumn (Doorenbos and Pruitt, 1977). To emulate the heat storing capacity of deep water the temperature function was averaged with the previous month, assigning a value of zero whenever ice was presumed to be present. Over the ice-free months, the crop factor was then modified by 20% of that of shallow water for $f(T)=0$ and $f(T)=1$ respectively., adjusting the values so that $\bar{k}_{c\ deep} = k_{c\ shallow}$ as expected on ground of energy conservation.

For the wetland areas, the maximum crop factor was set to 1.2 [-], the minimum varying with the minimum monthly temperature: for tropical regions, the value was set to 1.0, for temperate regions without frost to 0.6 and for regions with killing frost at 0.3 [-] based on values proposed by Allen et al. (1998). Between these extremes, the crop factor was distributed on the basis of the temperature function $f(T)$, k_c being zero during the months with frost. To account for the effect of humidity and wind speed, the monthly values were adjusted by the meteorological factor of Equation 22, applying a typical vegetation height of 2 m for tropical regions and of 1 m elsewhere.

Updates to the handling of evapotranspiration within PCR-GLOBWB

The reference potential evapotranspiration and the crop factors derived from the CRU TS 2.1 dataset necessitate an update to the processing of the evapotranspiration in PCR-GLOBWBWB. In the model, a distinction is made between bare soil evaporation, which is drawn from the upper soil layer after deduction of any evaporation of liquid water stored in the snow cover, and transpiration by vegetation, which is drawn from both soil layers in proportion to the relative root volume present after deduction of any evaporation of intercepted rainfall. So far, mainly actual evapotranspiration, *e.g.*, from the ECMWF ERA-40

reanalysis, was imposed. This actual evapotranspiration was fractioned on the basis of vegetation cover only and merely limited by the availability of soil moisture in the pertinent layers.

The use of the reference potential evaporation in combination with the crop factor implies that no distinction between soil evaporation and transpiration is made; k_c ranges from a minimum value for bare soil evaporation, $k_{c\ min}$, to a maximum value during the height of the growing season, $k_{c\ full}$, dependent on the LAI (see Figure 2 and Equation 21). Hence, fractioning on the basis of the vegetation cover can be abandoned and the bare soil potential evaporation is taken to be equivalent to:

$$ES_0 = k_{c\ min} \cdot ET_0, \quad \text{Equation 24}$$

and the total potential transpiration to:

$$T_0 = ET_c - ES_0 = k_c \cdot ET_0 - k_{c\ min} \cdot ET_0 = (k_c - k_{c\ min})ET_0, \quad \text{Equation 25}$$

where ET_c is the monthly crop specific potential evapotranspiration, ET_0 the potential reference evapotranspiration, both in $[L \cdot T^{-1}]$ (Equation 1), and k_c and $k_{c\ min}$ [-], are the monthly crop factor and the minimum crop factor for bare soil evaporation respectively.

Reductions of the potential bare soil evaporation and the transpiration are directly or indirectly related to the available soil moisture storage. For the bare soil evaporation, no reduction is applicable for the saturated fraction, x , of each cell as obtained by the Improved Arno Scheme of Hagemann and Gates (2003), except that the rate of evaporation cannot exceed the saturated hydraulic conductivity of the upper soil layer. Likewise, the potential bare soil evaporation over the unsaturated area, $1-x$, is only limited by the unsaturated hydraulic conductivity of the upper soil layer:

$$ES = x \cdot \min(k_{s_1}, ES_0) + (1-x) \min(k(\theta_E)_1, ES_0). \quad \text{Equation 26}$$

The uptake of transpiration by plants depends on the total available moisture in the soil layers. The fraction between the actual and potential transpiration rate is given by:

$$f_t = \frac{1}{1 + (\theta_E / \theta_{E50})^{-3B}} \quad \text{Equation 27}$$

where θ_{E50} is the effective degree of saturation at which the potential transpiration is halved and B is the coefficient of the soil water retention curve according to Clapp and Hornberger (1978).

The lack of aeration prevents the uptake of water by root under fully saturated conditions. The average degree of saturation over the unsaturated fraction of the cell is estimated by:

$$\theta_E = \frac{w(x)}{w} = \frac{W_{max} + b(W_{max} - W_{min}) \left[1 - \frac{b+1}{b} \left(\frac{W_{max} - W}{W_{max} - W_{min}} \right)^{\frac{1}{b+1}} \right]}{W_{max} + b(W_{max} - W_{min}) \left[1 - \left(\frac{W_{max} - W}{W_{max} - W_{min}} \right)^{\frac{1}{b+1}} \right]}, \quad \text{Equation 28}$$

where w is the sub-grid distributed maximum moisture storage and $w(x)$ the actual storage corresponding to the fraction x of saturation, the uppercase symbols refer to the

minimum and maximum storage capacities and the cell-averaged storage (W_{min} , W_{max} and W respectively [m]) and b is the shape factor describing the distribution of w .

In the above, the root fraction, r_{frac} , is only considered in obtaining the parameters B and θ_{E50} of the scaling function, for which the soil water retention characteristics of layers i were weighed by both the root fraction and the maximum storage capacity per layer. The actual transpiration rate is partitioned over the upper two soil layers on the basis of the relative storage $W_i/\Sigma W_i$ by local soil water availability, if applicable.

Downscaling CRU monthly values to daily model input

Because of the presence of non-linear processes like interception, the monthly fields of the CRU TS 2.1 have to be downscaled to daily fields in order to force PCR-GLOBWB successfully. The required input consists of precipitation, temperature and potential evapotranspiration which were broken down by means of daily fields of temperature and precipitation as obtained from the ERA-40 reanalysis. The ERA-40 re-analysis was preferred as this provides a high-quality global analysis of the atmospheric conditions over the past four decades (Uppala et al., 2005) regardless of errors in the absolute errors like the severe overestimation of precipitation over tropical areas (Troccoli and Källberg, 2004). In this manner, the CRU forcing is directly applicable to the period 1958-2001 for which calendar years actual ERA-40 results are available. Prior to 1958, ERA-40 years should be used as proxy for the then present atmospheric conditions and selected on the basis of annual temperature and precipitation over larger areas, *e.g.*, continents, as witnessed by the CRU TS 2.1 dataset. The daily fields of precipitation and temperature were obtained from six 6-hourly forecasts from 12 hours UTC onwards where the first two forecasts were removed to avoid any bias due to spin-off effects. Thus, 24-hour forecasts were obtained from 00 hours UTC onwards, being the daily sum of convective and large-scale precipitation for rainfall and the average 2m-air temperature for the temperature.

For consistency, the forcing complied entirely with the CRU dataset, the aggregated daily values returning the monthly statistic, although one could argue that the ERA-40 daily temperature fields are more likely to approximate the actual conditions. In the case of temperature, a simple additional anomaly was used:

$$T = T_{ERA} + (\bar{T}_{CRU} - \bar{T}_{ERA}), \quad \text{Equation 29}$$

where T denotes the daily temperature [$^{\circ}\text{C}$] and \bar{T} the mean monthly value for the ERA-40 and CRU datasets.

To obtain daily reference potential evapotranspiration, E_0 [$\text{m}\cdot\text{day}^{-1}$], the monthly value was multiplied with the normalized ERA-40 daily temperature, which was transformed to Kelvins to take care of freezing:

$$E_0 = \bar{E}_0 \frac{T_{ERA} + 273.2}{\bar{T}_{ERA} + 273.2}. \quad \text{Equation 30}$$

For the precipitation the preferred anomalies were also multiplicative:

$$P = P_{ERA} \frac{\bar{P}_{CRU}}{\bar{P}_{ERA}}. \quad \text{Equation 31}$$

Although \overline{P}_{ERA} is always greater than zero, severe rounding errors may occur whenever it is very small. To overcome this problem, a threshold was defined as the mean daily precipitation for the present month according to the CRU dataset that had to be exceeded for the multiplicative anomaly to be used:

$$P_{crit} = \frac{\overline{P}_{CRU}}{\overline{W}_{CRU}}, \quad \text{Equation 32}$$

where \overline{W}_{CRU} is the monthly number of wet days.

If the threshold was not exceeded, as might be expected in some arid regions, a daily rainfall estimate was used for a limited number of days. First, a temperature limit for the ERA-40 re-analysis was estimated by linear interpolation:

$$T_{crit\ ERA} = T_{min\ ERA} + (T_{max\ ERA} - T_{min\ ERA}) \frac{\overline{W}_{CRU}}{N}, \quad \text{Equation 33}$$

where T_{max} and T_{min} are the maximum and minimum daily ERA-40 temperature for a given month, \overline{W}_{CRU} is the monthly number of wet days in the CRU dataset and N is the total number of days in the current month.

For each month the number of days that the ERA-40 temperature falls below this threshold is calculated as an approximation of the wet days, \overline{W}_{ERA} , and if the temperature is evenly distributed this will approximate \overline{W}_{CRU} . The corresponding daily rainfall is applied for each day that $T_{ERA} < T_{crit\ ERA}$ and calculated as:

$$\hat{P} = \frac{\overline{P}_{CRU}}{\overline{W}_{ERA}}. \quad \text{Equation 34}$$

Results

CRU timeseries and climatology

Of all variables contained by the CRU TS 2.1 dataset that are directly or indirectly required by PCR-GLOBWB, the mean annual values for the climatology 1961-1990 are presented in Figure 3.

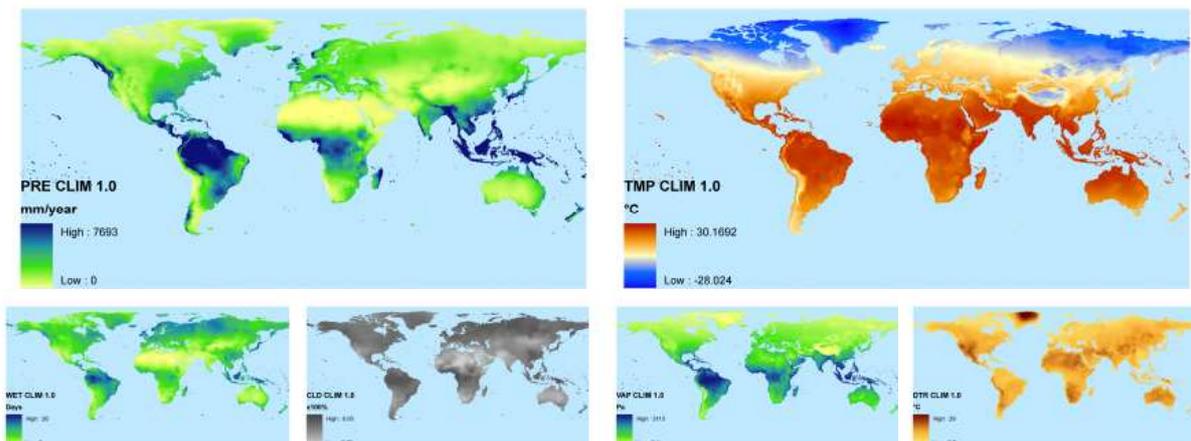


Figure 3: Mean annual values of CRU variables of interest for the climatology 1961-1990. Top row: precipitation (PRE, mm per year) and temperature (TMP, °C); bottom row: number of wet days (days per year), cloud cover (fraction), vapour pressure (Pa) and daily temperature range (°C).

For the primary variables of precipitation and temperature also temporal variability was considered. Over the full period of the CRU TS 2.1 dataset monthly values were summed to annual totals. From these annual totals, the mean and standard deviation were calculated for three different periods: i) the total length of the CRU TS 2.1 series, ii) the reference climatology period 1961-1990, and iii) the period 1958-2001, covering most of the ERA-40 reanalysis period. These periods were subsequently compared by means of Student's t-test where the t-statistic was calculated as:

$$t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}, \quad \text{Equation 35}$$

where \bar{x} stands for the mean, s^2 for the variance and n for the number of years for the sub-periods 1 and 2 respectively. t is the test statistic with n_1+n_2-2 degrees of freedom.

This test statistic was calculated for each cell with respect to the climatology and binned for four absolute significance levels (0.01, 0.05, 0.10 and 0.25 two-tailed; Figure 4).

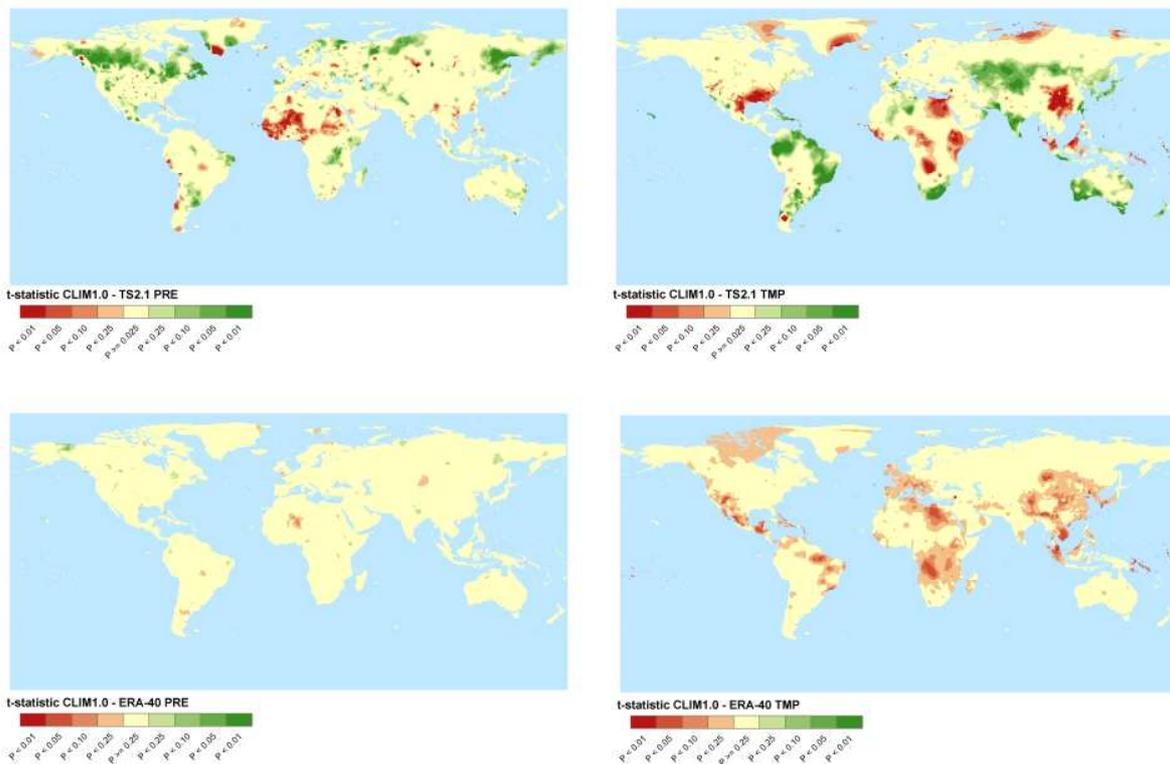


Figure 4: Significance of differences in mean between the climatology over 1961-1990 and the total CRU TS 2.1 period 1901-2002 (top row) and the ERA-40 sub-period 1958-2001 (bottom row) for precipitation (left) and temperature (right). Red colours reflect that the climatology has a lower mean, green colours that it has a higher mean than the comparison period.

For the entire globe and all continents the variation in precipitation and temperature was calculated over the full length of the CRU TS 2.1 period (1901-2002; Figure 5). For the

climatology over the years 1961-1990, also the zonal distribution of both variables was calculated for each season (Figure 6)

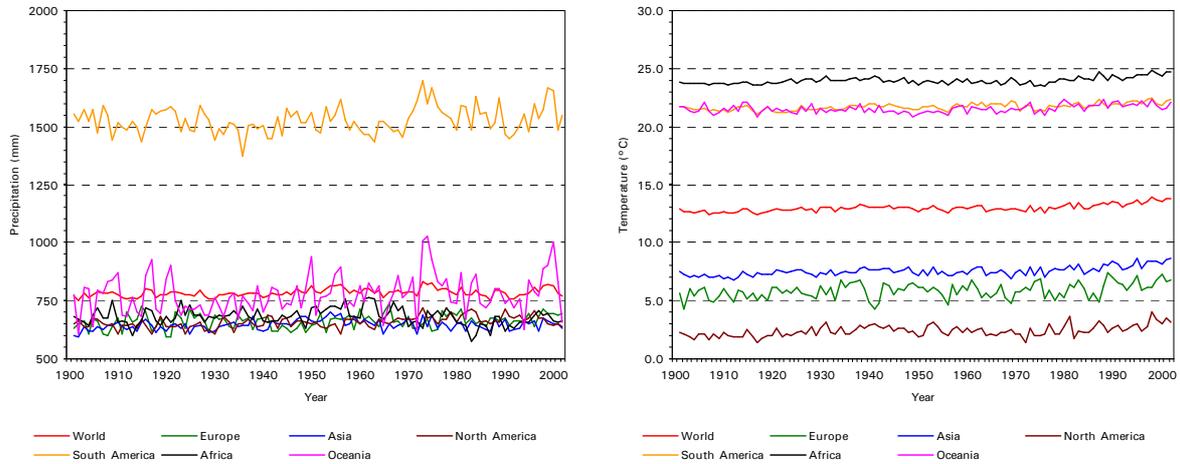


Figure 5: Variations in annual global and continental precipitation (left) and temperature (right) over the CRU TS 2.1 period 1901-2002.

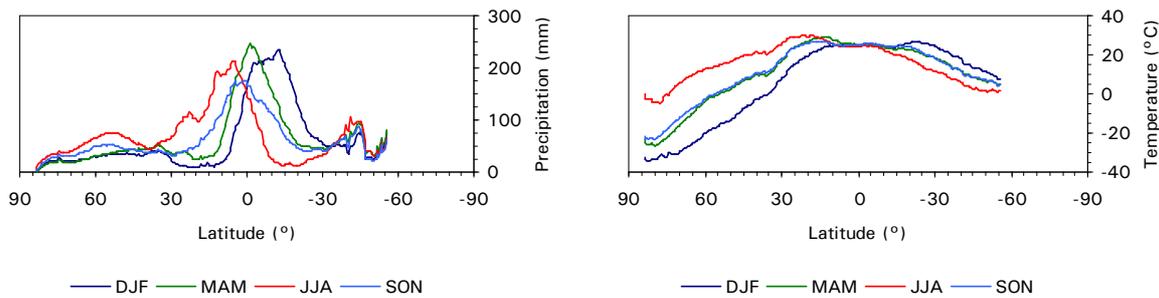


Figure 6: Seasonal variation in precipitation (left) and temperature (right) per half-degree zone for the 1961-1990 climatology: DJF (December-January-February), MAM, JJA, and SON.

To evaluate the discrepancy between the ERA-40 reanalysis and the CRU TS 2.1 dataset, the difference between the two datasets has been calculated for the mean annual precipitation and temperature over the period 1958-2001 (Figure 7).

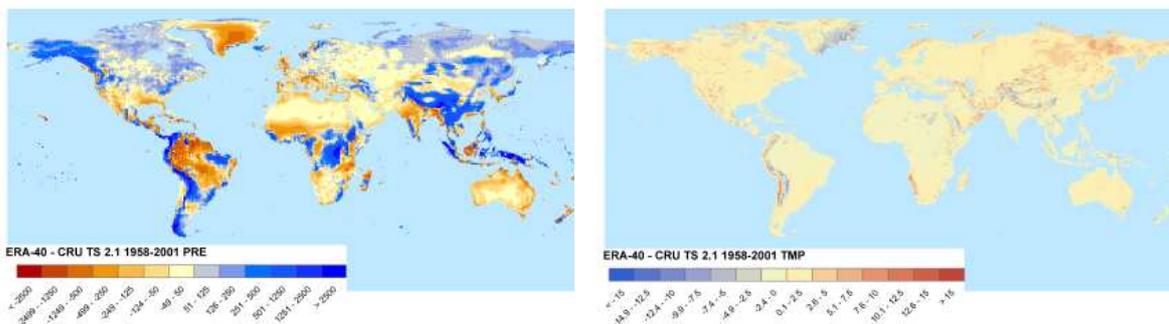


Figure 7: Difference fields for precipitation (left), and temperature (right) between the ERA-40 reanalysis and CRU dataset over the period 1958-2001.

Holdridge life zone classification

The Holdridge life zone classification calculated from the CRU climatology over the years 1961-1990 can be compared to the global coverage generated by Leemans (1989) on the basis of a dataset of 5500 stations with monthly temperature and precipitation, distributed over the world and covering between 10 and 40 years within the preferred period of 1931-1960 (Leemans, 1990). The aggregated dataset was used that combines the 39 Holdridge life zone classes into 14 larger entities (Cold Parklands; Forest Tundra; Boreal Forest; Cool Desert; Steppe; Temperate Forest; Hot Desert; Chaparral; Warm Temperate Forest; Tropical Semi-Arid; Tropical Dry Forest; Tropical Seasonal Forest; Tropical Rain Forest). A cross-table of this map with the CRU-based classification was made and the extended CRU classification re-classified accordingly. In total, 44% of the land surface had no consistent match in Leemans' aggregated life zones classification, with 7% of the mismatch attributable to a difference in land mask. The poorest agreement concerned the polar zones and some tropical regions, due to differences in temperature and precipitation over these areas (Figure 8).

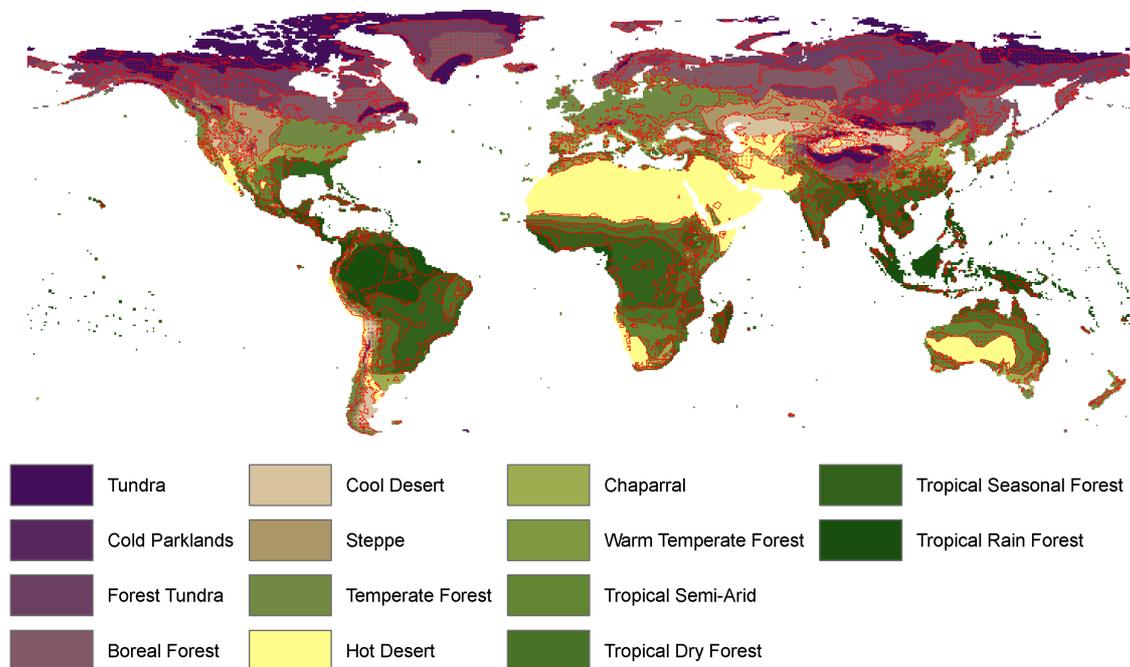


Figure 8: Aggregated Holdridge Life zones classification according to Leemans (1989) for the CRU 1961-1990 climatology. Areas without a consistent match are shaded.

Reference potential evapotranspiration

The monthly reference potential evapotranspiration was calculated for the period 1901-2002 covered by the CRU TS 2.1 dataset and summed to yield the annual total. Similar to the primary variables of precipitation and temperature, the differences between the three main periods were tested for their significance (Figure 9).

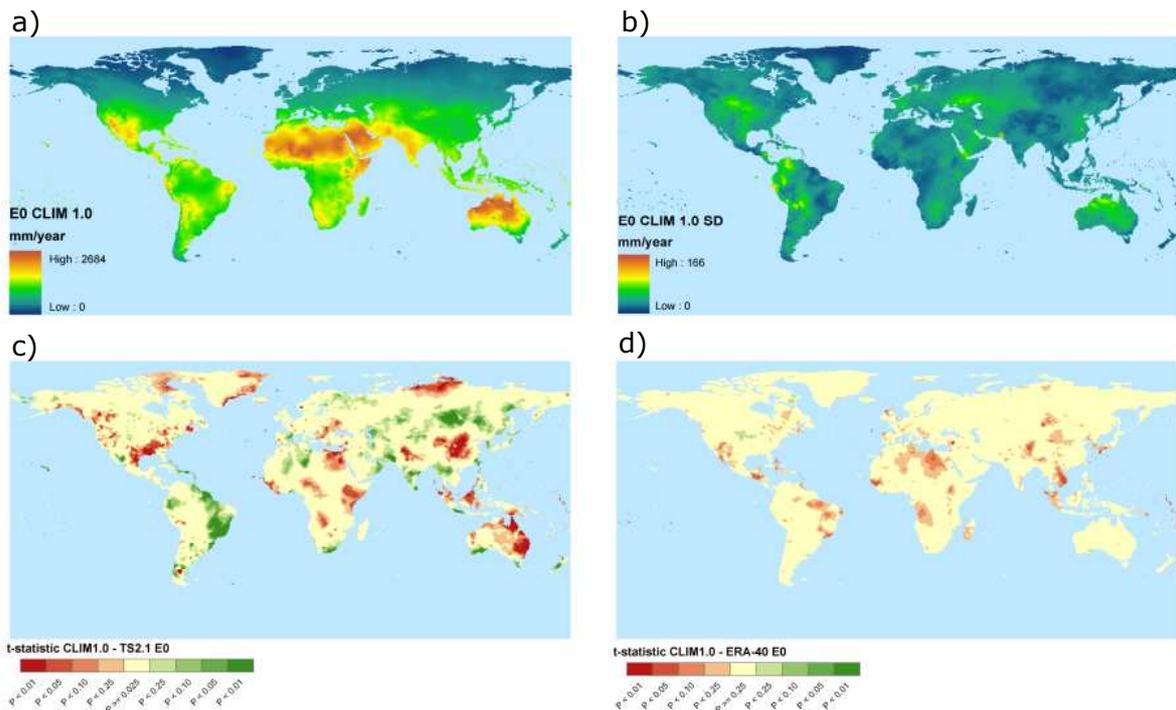


Figure 9: a) Mean and b) standard deviation in reference potential evapotranspiration E_0 over the period 1961-1990 (CRU baseline climate); c) significance of differences in mean E_0 over the entire CRU TS 2.1 period (1901-2002) and d) the period of the ERA-40 reanalysis (1958-2001).

For the entire globe and all continents the variation in the reference potential evapotranspiration, E_0 , was equally calculated over the full length of the CRU TS 2.1 period (1901-2002; Figure 10.a). For the climatology over the years 1961-1990, also the zonal distribution of E_0 was calculated for each season.

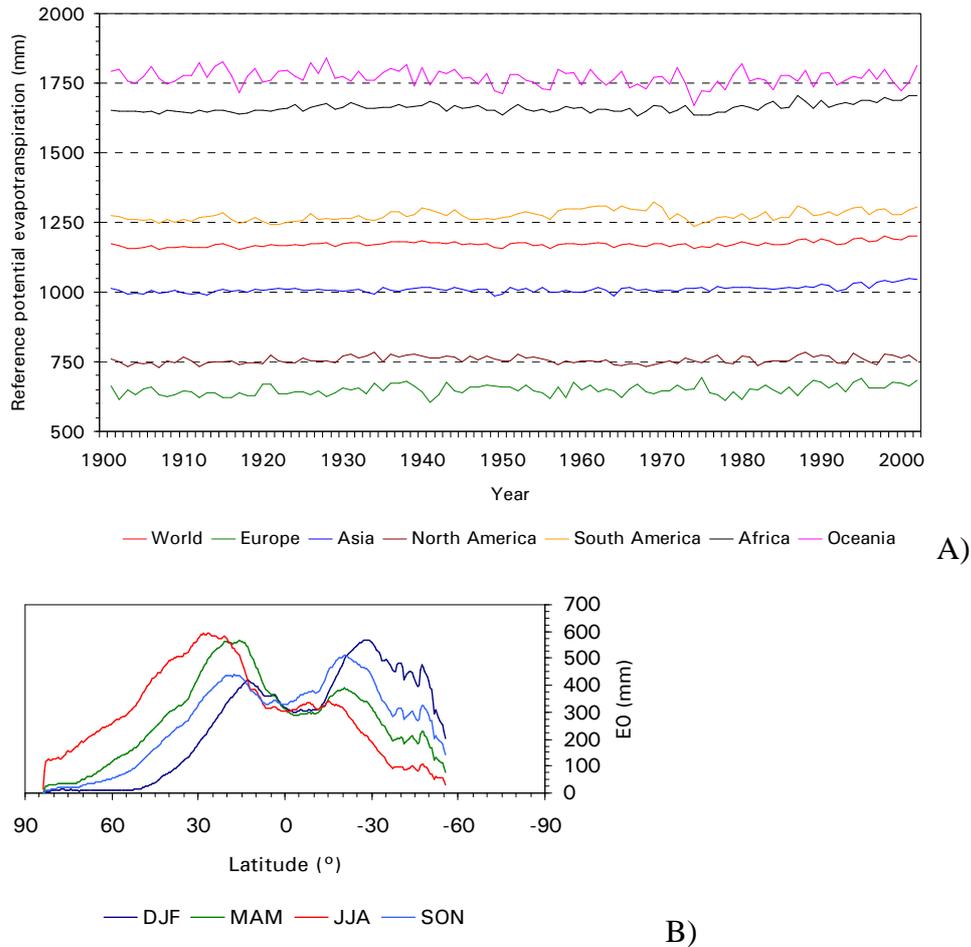


Figure 10: Variations in annual global and continental reference potential evapotranspiration, E_0 , over the CRU TS 2.1 period 1901-2002; and, b) seasonal variation in E_0 per half-degree zone for the 1961-1990 climatology: DJF (December-January-February), MAM, JJA, and SON.

No direct comparison was possible between the CRU and ERA-40 evapotranspiration as the former was calculated as reference potential evapotranspiration in this study and the latter was reported as the actual evapotranspiration from the ECMWF land surface model.

Crop factors

Crop factors were calculated for the GLCC dataset as described by the methodology above. The actual crop factor was based on the vegetation characteristics linked to the GLCC dataset and the meteorological correction factor of Equations 20 and 22. The length of the growing season could be constrained by temperature or moisture availability. Figure 11 shows the global distribution of the estimated crop factor on the basis of the water balance of Equation 14. It identifies the major arid regions of the world as well as regions with severe seasonal water shortage such as the Mediterranean, part of the Asian monsoon-belt (*e.g.*, India), subtropical South America, and the Arctic deserts (*cf.* Figure 8). This crop factor not only influences the length of the growing season but also the rate at which the vegetation transforms from one stage to the other (Figure 2).

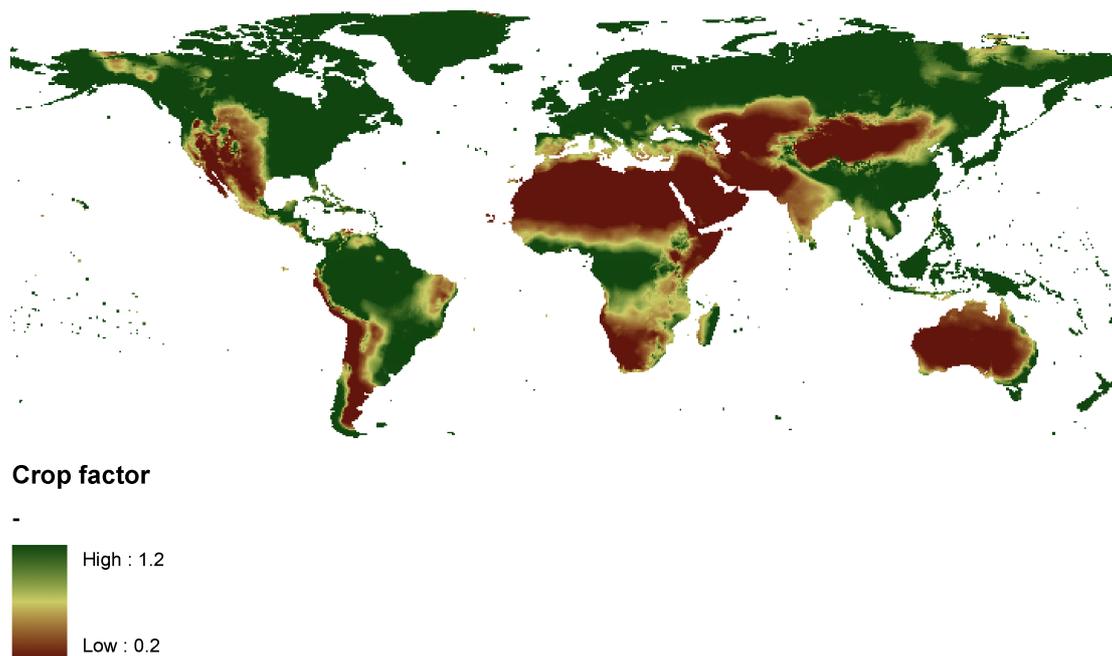


Figure 11: Crop factor optimized for moisture availability.

The length of the entire growing season and the optimum season are shown in Figure 12. Except for the arctic and some arid regions, most areas reveal a considerable growing season. The length of the growing season with optimum conditions is much lower, especially in temperate areas where temperature is the limiting factor or in semi-arid or arid regions, where moisture is a limiting factor. Some seasonality is also present over the tropics. For Africa, this is also shown by the Holdridge classification (Figure 8). It is less widespread, however, in South-America, and seasonality may be introduced here as the optimized crop factor of Figure 11 that is at its maximum value. The influence of seasonality on growth is equally manifested by the seasonal relative LAI (Figure 13).

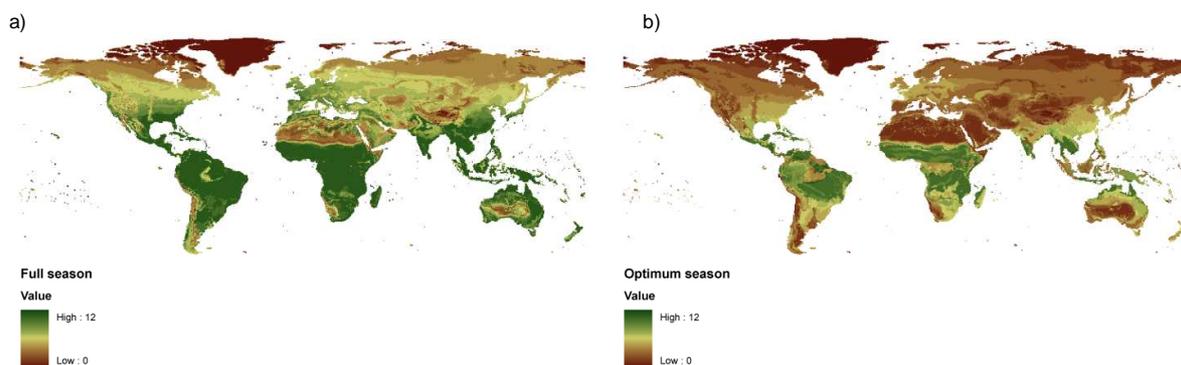


Figure 12: Length of growing season (months per year): a) full season, b) optimum season.

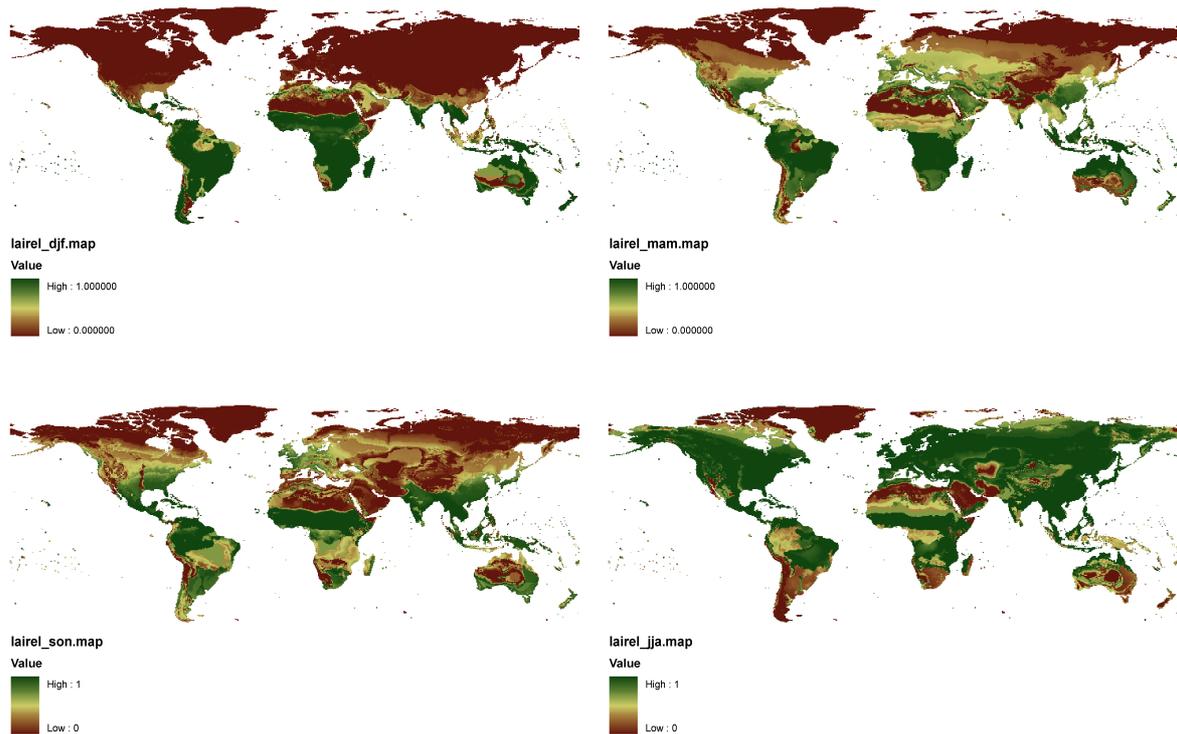


Figure 13: Seasonal variation in relative LAI. From topleft, clockwise: DJF, MAM, JJA, SON

The actual crop factor ranges between the minimum of 0.2 [-] for bare surfaces to the maximum that depends on height (Equation 20). Theoretically, the maximum value is limited to 1.2 [-] but this value can increase as result of the local meteorology (Equation 21; Figure 14). Increased values are found above areas that have a low relative humidity or high wind speeds. Areas with increased crop factors are present over the semi-arid regions of North America, the Mediterranean, the Sahara, Central Asia and Australia. The largest increase are found over Greenland and Spitsbergen and northwest South-America. As there is no growing season in the polar region the elevated values are of no direct concern. In South-America the higher values are the result of low vapour pressures (Figure 3) with exceptional high wind speeds of over $20 \text{ m}\cdot\text{s}^{-1}$ during summer. Once corrected for the local meteorology, maximum crop factors for tall vegetation reach a maximum value of 1.26. For the short vegetation type, the maximum value is 1.13 as the shorter vegetation height yields a lower maximum crop factor to start with (Equation 20; Appendix 4). The minimum and maximum values of the crop factors of short and tall vegetation, aggregated from 30 arc sec to 0.5° , are shown in Figure 15. Overall, this picture agrees well with the expectation that tall vegetation will have larger crop factors as this vegetation type is generally taller and perennial, thus having less seasonal variation than short vegetation types that include annuals like true crops. There is no increase in crop factors over true deserts covered by short vegetation only. Remarkably, some areas with lower vegetation of the same signature are found in other areas, like the northeast of the USA and the Congo basin. Partly, this corresponds with urban areas or with wrongly-classified cells (*e.g.*, deserts occurring within the rain forest of the Congo basin). In general, these areas are small, covering less than 1% of each cell, and can be expected to have little influence on the total outcome of the model.

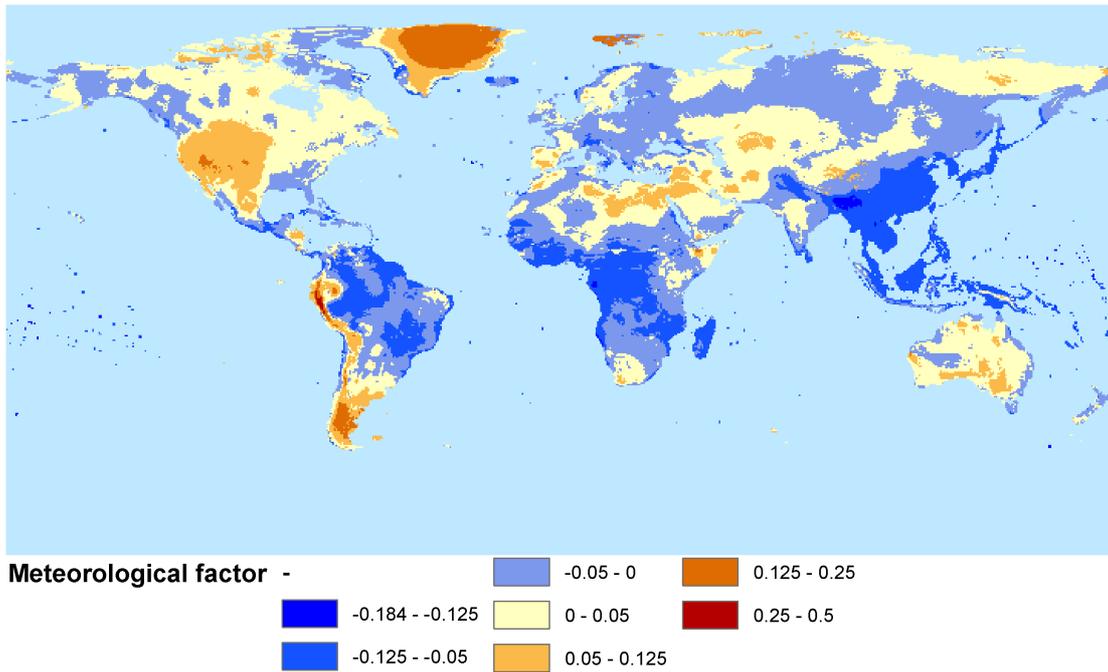


Figure 14: Meteorological correction factor of Equation [X] for $(\frac{h}{3})^{0.3} = 1$.

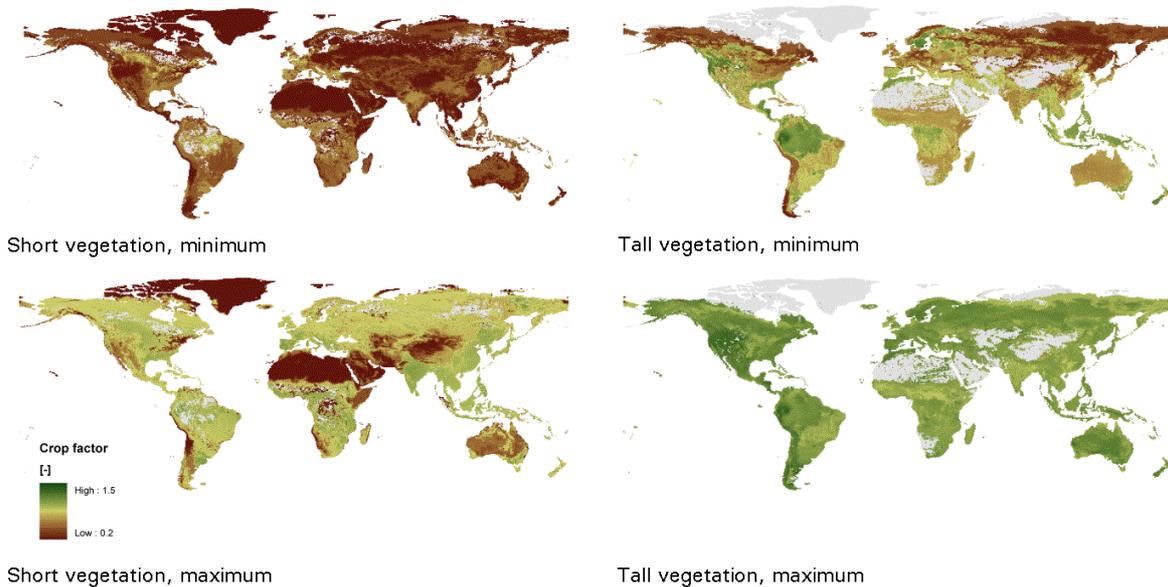


Figure 15: Minimum and maximum values of the crop factor for short (left) and tall vegetation as obtained for the GLCC dataset. Grey areas indicate regions where a particular type is absent.

Open water evapotranspiration is modified by temperature only, heat storage effects leading to a shift in time compared to the air temperature. The maximum crop factor amounts to 1.48 in dry areas, the minimum is zero in regions where seasonal ice covers occur.

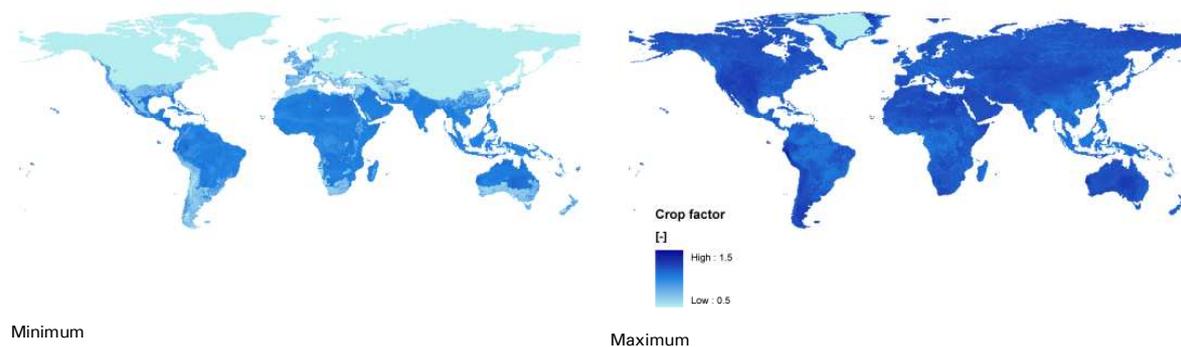


Figure 16: Minimum and maximum crop factors for open water evaporation

Conclusions

This report describes in detail the parameterization and particularities of PCR-GLOBWB related to CRU forcing. Emphasis is placed on evapotranspiration, as this requires the formulation of crop factors and the calculation of reference potential evapotranspiration according to the FAO guidelines (Allen et al., 1998). Following the land surface division in PCR-GLOBWB, crop factors have to be defined for three surfaces (i.e., short and tall vegetation and open freshwater) and over time. The basic assumption that natural vegetation and planted crops strive to maximize the available resources underlie the definition of the crop factors. Temperature and moisture availability limit the available growing season and is relative only. The influence of the actual vegetation, as it arises from local conditions as nutrient availability etc., is taken into account through the parameterization of the LAI and vegetation heights as associated with the GLCC dataset.

Large differences exist between the ERA-40 and CRU dataset and between the different periods within the CRU dataset itself (i.e., climatology or full time series). In term of significance, temperature appears to differ more often than precipitation. However, temperature is more constant and significance in the difference-of-means test is lost due to the large variability of rainfall. Particular over tropical areas the ERA-40 dataset predicts much more precipitation than the CRU whereas the trend is reversed over temperate regions. Differences within the CRU dataset arise from changes in the density of the observation network (Congo, South America) but some changes might be attributed to different degrees of activity of large-scale climate phenomena as the ENSO over the respective periods (e.g., south of USA). Earlier parts of the CRU TS 2.1 should therefore be used with care.

In general, the CRU climatology leads to a consistent vegetation distribution as indicated by the Holdridge classifications. Discrepancies between the original dataset of Leemans (1989) and the CRU-based distribution mainly arises as a result of an inclusion of more stations and a different incorporation of surface elevation.

The crop factors from the CRU fall within the range of reasonable estimates as listed by Allen et al. (1998) and Doornkamp and Pruitt (1977). Also, there is a good agreement with reported growing seasons and phases. Since these crop factors partly alleviate the differences in the reference potential evapotranspiration as found for the different periods of the CRU database, the CRU-derived climatology of crop factors can be applied to the entire CRU TS 2.1 period, provided it can be assumed land cover remains unaltered, although this may result in an underestimation of the variability in actual evapotranspiration for the earlier part of the CRU period.

References

- Allen, R.G., Pereira, L.S., Raes, D., Smith, M. (1998). Crop evapotranspiration. FAO Irrigation and drainage paper 56. FAO, Rome.
- Clapp, R.B. and Hornberger, 1978. G.M. Empirical Equations for Some Soil Hydraulic Properties. *Water Resources Research* 14, 601-604.
- Dickinson, R.E., Henderson-Sellers, A., Kennedy, P.J. (1983). Biosphere-Atmosphere Transfer Scheme (BATS). NCAR technical note, Boulder.
- Dingman, S.L. (1994). *Physical Hydrology*. McMillan, New York.
- Doorenbos, J., Pruitt, W.O. (1977). Crop water requirements. FAO Irrigation and drainage paper 24. FAO, Rome.
- Hagemann, S., Botzet, M., Dümenil, L., and Machenhauer, B., 1999 Derivation of global GCM boundary conditions from 1 km land use satellite data *Max-Planck-Institute for Meteorology*, Report **289**, Hamburg, Germany. (Report available electronically from: <http://www.mpimet.mpg.de/deutsch/Sonst/Reports/HTMLReports/289/>)
- Hagemann, S. & Gates, L.D., 2003. Improving a subgrid runoff parameterization scheme for climate models by the use of high resolution data derived from satellite observations. *Climate Dynamics* 21, 349–359.
- Leemans, R., 1989. Global *Holdridge Life Zone Classifications*. Digital Raster Data on a 0.5-degree Geographic (lat/long) 360x720 grid. Laxenberg, Austria: IIASA. Available from http://www.ngdc.noaa.gov/ecosys/cdroms/ged_iiia/datasets/a06/lh.htm#top
- Leemans, R. 1990. Possible changes in natural vegetation patterns due to global warming. IIASA Working Paper WP90-08 and Publication Nr 108 of the Biosphere Dynamics Project. Laxenburg, Austria.
- Mitchell, T.D. and P.D. Jones 2005. An improved method of constructing a database of monthly climate observations and associated high-resolution grids. *International Journal of Climatology*, 2005. 25(6): p. 693-712.
- New, M., Hulme, M., Jones, P. (1999). Representing twentieth-century space-time climate variability. Part I: Development of a 1961-1990 mean monthly terrestrial climatology. *Journal of Climate* 12, 829-856.
- New, M., Hulme, M., Jones, P. (2000). Representing twentieth-century space-time climate variability. Part II: Development of 1901-96 monthly grids of terrestrial surface climate. *Journal of Climate* 13, 2217-2238.
- New, M., Lister, D., Hulme, M., Makin, I. (2002). A high-resolution data set of surface climate over global land areas. *Climate Research*. Vol. 21, 1-25.
- Olson, J.S., (1994a). Global ecosystem framework-definitions: USGS EROS Data Center Internal Report, Sioux Falls.
- Olson, J.S., (1994b). Global ecosystem framework-translation strategy: USGS EROS Data Center Internal Report, Sioux Falls.
- Troccoli, A. and P. Källberg, 2004. Precipitation correction in the ERA-40 reanalysis, in ERA-40 Project Report Series 13. 2004, ECMWF: Reading. p. 10.
- Uppala, S.M. *et al.* 2005, The ERA-40 re-analysis. *Quarterly Journal of the Royal Meteorological Society*, 2005. 131: p. 2961-3012.

Appendix 1: Code of cru_grid2asc.f

```

PROGRAM CRUTSS
C
C   IMPLICIT NONE
C   Reads in new 0.5° CRU global gridded dataset containing the years 1901-2000
C   and converting them to individual ascii files for the surface fields
C   specified in
C   VARID. All data are processed as are. RvB, UU, April 2008
C
C   INTEGER NCOLS,NROWS,NMONTHS
C   PARAMETER (NCOLS= 720,NROWS= 360,NMONTHS= 12)
C   INTEGER STARTYEAR, ENDYEAR, NVAR
C   PARAMETER (STARTYEAR= 1901, ENDYEAR= 2002, NVAR= 1)
C   INTEGER MONTHINDEX, YEARINDEX, VARINDEX
C   INTEGER XCNT, YCNT
C   INTEGER IVAL(NCOLS,NROWS,NMONTHS)
C   CHARACTER*(*) PRFX, SFFX
C   PARAMETER (PRFX= "cru_xxx",SFFX= '.asc')
C   CHARACTER*30 GRIDFILE,INFILE
C   CHARACTER*18 ASCFILE
C   CHARACTER*2 MONTHSTR
C   CHARACTER*3 VARID(NVAR),VARSTR
C   CHARACTER*4 YRSTR
C
C   DATA VARID /"wet","vap","tmp"/
C   DATA VARID /"dtr"/
C   GRIDFILE="cru_ts_2_10.1901-2002.xxx.grid"
C   DO VARINDEX=1,NVAR
C     VARSTR= VARID(VARINDEX)
C     INFILE= GRIDFILE
C     INFILE(23:25)= VARSTR
C     OPEN(UNIT=1,FILE= INFILE,STATUS="OLD",ERR= 999)
C     DO YEARINDEX= STARTYEAR,ENDYEAR
C       CALL STRCONV(YEARINDEX,YRSTR,4)
C       READ (1,*) IVAL
C       DO MONTHINDEX= 1,NMONTHS
C         CALL STRCONV(MONTHINDEX,MONTHSTR,2)
C         WRITE (*,'(4X,A3,1X,I4,1X,I2)') VARSTR,YEARINDEX,MONTHINDEX
C         ASCFILE= PRFX//YRSTR//"_ "//MONTHSTR//SFFX
C         ASCFILE(5:7)= VARSTR
C         OPEN(UNIT=2,FILE= ASCFILE,STATUS= "REPLACE",ERR= 998)
C         DO YCNT= NROWS,1,-1
C           DO XCNT= 1,NCOLS
C             WRITE (2,'(I5,1X,$)') IVAL(XCNT,YCNT,MONTHINDEX)
C           END DO
C           WRITE (2,*)
C         END DO
C         CLOSE(UNIT=2,STATUS= "KEEP")
C       END DO
C     END DO
C     CLOSE(UNIT=1)
C   END DO
C   GOTO 999
998  WRITE (*,'(2X,"Program terminated with error")')
C   GOTO 1000
999  WRITE (*,'(2X,"Program terminated")')
1000 END
C
C *****
C SUBROUTINE STRCONV(XVAL,XSTR,DIGITS)
C
C DECLARATION
C
C INTEGER XVAL, CNT, DIGITS, ID1, ID2
C CHARACTER*(4) XSTR
C CHARACTER*1 CD

```

```

C
C   STEPPING THROUGH DIGITS
      ID2= 0
      XSTR= ''
      DO CNT= 1,DIGITS
        ID1= ID2
        ID2= INT(XVAL*0.1**(DIGITS-CNT))
        ID1= ID1*10
        CD= ACHAR(INT(ID2-ID1)+48)
        XSTR(CNT:CNT)= CD
      END DO
C
      RETURN
C
      END

```

Python script mapgen_cru.py

#Converts ascii files from the CRU TS1 dataset to pcraster maps

```

import os, zlib, zipfile
import pcraster as pcr

def zipToDir(file, dir):
    try:
        os.mkdir(dir, 0777)
    except:
        pass
    zfobj = zipfile.ZipFile(file)
    for name in zfobj.namelist():
        if name.endswith('\\'):
            try:
                os.mkdir(os.path.join(dir, name))
            except:
                pass
        else:
            try:
                outfile = open(os.path.join(dir, name), 'r')
                outfile.close()
            except:
                outfile = open(os.path.join(dir, name), 'w')
                outfile.write(zfobj.read(name))
                outfile.close()

def ascToMap(FileNameIn,FileNameOut,CloneFile):
    command = 'asc2map %s %s --clone %s -S -m -999' %
(FileNameIn,FileNameOut,CloneFile)
    os.system(command)

#declaration: parameter suffices of input files and number of parameters
ParID= ['dtr'] # 'pre','cld','tmp','vap','wet'] #
#folder with data
PathCRU= 'i:\\cru05_timeseries\\newlygridded'
#start and end year of sequence
StartYear= 1901
EndYear= 2002
NMonths= 12
#Clone map definition
CloneFile= '..\\..\\globalclone.map'

print 'PROCESSING OF CRU TS1 dataset of monthly global meteorological fields'
for Par in ParID:
    print '\\tProcessing %s' % Par
    #sets input path and tests for the existence of last written data file after
    #checking for existing zip-files and extracting any non existing data
    print '\\tchecking for availability of non-processed data in zip-files'
    PathIn= PathCRU
    if os.path.exists(PathIn):
        os.chdir(PathIn)

```

```

else:
    break
PathOut= os.path.join(PathIn,Par)
try:
    FileNameIn= 'cru05_%s_asc.zip' % (Par)           #archive with monthly ascii
files
    zipToDir(FileNameIn,PathOut)
    FileNameIn= 'cru05_%s.zip' % (Par)             #archive with monthly
pcraster maps
    zipToDir(FileNameIn,PathOut)
except:
    print '\t\tno exisiting zipfiles found'
    pass
    #opening zip files for output
#-monthly text and map files, ascii output and stack
FileNameOut= 'cru05_%s_asc.zip' % (Par)           #archive with monthly ascii
files
AsciiZip= zipfile.ZipFile(FileNameOut,'w',zipfile.ZIP_DEFLATED)
FileNameOut= 'cru05_%s.zip' % (Par)             #archive with monthly pcraster maps
StackZip= zipfile.ZipFile(FileNameOut,'w',zipfile.ZIP_DEFLATED)
#loop for years
PathIn= PathOut
for YearOrd in range(StartYear,EndYear+1):
    #processing files if file of interest found
    print '\tconverting ASCII to CSF'
    print '\t\t%s' % YearOrd
    if os.path.exists(PathIn):
        os.chdir(PathIn)
    else:
        break
    FileNameCheck= 'cru_%s%04d_%02d.asc' % (Par,EndYear,NMonths)
    CheckFileExists= 0
    while CheckFileExists == 0:
        try:
            cf= open(FileNameCheck,'r')
            CheckFileExists= 1
            cf.close()
        except:
            CheckFileExists= 0
    #=> continuation when file found
    #generating maps for present variable
    #loop for months
    for MonthOrd in range (1,NMonths+1):
        #filenames
        FileNameIn= 'cru_%s%04d_%02d.asc' % (Par,YearOrd,MonthOrd)
        FileNameOut= 'cru_%s%04d.%03d' % (Par,YearOrd,MonthOrd)
        #conversion
        CheckFileExists= 1
        try:
            cf= open(FileNameIn,'r')
            cf.close()
        except:
            CheckFileExists= 0
        if CheckFileExists== 1:
            ascToMap(FileNameIn,FileNameOut,CloneFile)
        #zipping files
        try:
            AsciiZip.write(FileNameIn)
            StackZip.write(FileNameOut)
        except:
            pass
    #closing zips
    AsciiZip.close()
    StackZip.close()
#removing files in present path
for Root, Dirs, Files in os.walk(PathIn):
    if len(Dirs) > 0:
        print 'subdirectories present in %s !' %(Root)

```

```
        break
    else:
        for FileCount in range(0,len(Files)):
            FileNameOut= Files[FileCount]
            os.remove(FileNameOut)
```

Appendix 2: Python script cru_proc.py

```
# Processes CRU Timeseries and Climatology to derive:
# 1) climatology over the years 1961-1990
# 2) detect any change in the mask over the CRU timeseries
# 3) create a patch for any area missing between the climatology and timeseries
# 4) update the input where necessary (done internally) and
#     calculate the reference potential evapotranspiration over the globe
# 5) return statistics
# 6) calculate temporal course of water stress and temperature in order to
determine growing season
# 7) calculate temporal statistics of annual precipitation, temperature and
evapotranspiration
#RvB, UU, June 2008
#includes new parameterization of the growing season

import os, zlib, zipfile,shutil,calendar
import pcraster as pcr

def zipToDir(file, dir, a_name):
    #extracts all existing file from directory
    try:
        os.mkdir(dir, 0777)
    except:
        pass
    zfobj = zipfile.ZipFile(file)
    for name in zfobj.namelist():
        if name.endswith('/'):
            try:
                os.mkdir(os.path.join(dir, name))
            except:
                pass
        else:
            try:
                outfile = open(os.path.join(dir, name), 'r')
                outfile.close()
            except:
                outfile = open(os.path.join(dir, name), 'w')
                outfile.write(zfobj.read(name))
                outfile.close()

def generateNameT(RootName,Number):
    #generates file name akin to old pcraster py generateNameT
    #-split RootName into different elements, keeping root of file name only
    # to expand file name and convert Number to string: takes up to 8 digits
    timesteps
    NumStr= str(int(Number))
    NrChar= len(NumStr)
    if NrChar>8:
        print 'number exceeds allowable bounds'
    else:
        FileNameList= (RootName.split('/'))
        FileNameIn= FileNameList[len(FileNameList)-1][0:11-NrChar]
        while len(FileNameIn)<8:
            FileNameIn= FileNameIn+'0'
        FileNameIn= FileNameIn+'000'
        if len(NumStr)<3:
            while len(NumStr)<3:
                NumStr= '0'+NumStr
        NrChar= len(NumStr)
        FileNameOut= ''
        for icnt in range(0,12-NrChar-1):
            FileNameOut= FileNameOut+FileNameIn[icnt]
        for icnt in range(0,NrChar):
            FileNameOut= FileNameOut+NumStr[icnt]
        FileNameIn= FileNameOut
        FileNameOut= ''
        for icnt in range(0,8):
            FileNameOut= FileNameOut+FileNameIn[icnt]
```

```

        FileNameOut= FileNameOut+'.'
        for icnt in range(8,11):
            FileNameOut= FileNameOut+FileNameIn[icnt]
        for NrChar in range(len(FileNameList)-2,-1,-1):
            FileNameOut= FileNameList[NrChar]+'/'+FileNameOut
        return FileNameOut

# MAIN
# spatial attributes and global options
CloneMap= 'maps/globalclone.map'
pcr.setclone(CloneMap)
pcr.setglobaloption('radians')
cruClone= pcr.cover(pcr.readmap(os.path.join('maps','cru_clone.map')),0)
# duration of total timeseries
NMonths= 12
StartYear= 1901
EndYear= 2002
# CRU timeseries parameters and conversion factors
# (not used on original maps and derived climatology)
# to: millimeter,degree Celsius, percentage, hPa and days
CRUArchivePath= '../..../archive/crutss'
CRUTSSPath= 'newlygridded'
CRUClimPath= 'climatology'
CRUGrowthPath= 'growingseason'
CRUPeriodPath= 'stats'
CRUTSSParamList= ['tmp','pre','cld','vap','wet','dtr']
CRUTSSConvList= [0.1,0.1,0.1,0.1,0.1,0.01,0.1]
ClimStart= 1961
ClimEnd= 1990
JulianDay= [1,32,60,91,121,152,182,213,244,274,305,335,366]
# 0) Start
print 'Processing CRU climatology and timeseries'
    1) deriving climatology
print '\t%s' % 'calculating climatology over 1961-1990 for:'
ParamCnt= 0
CRUTSSMask= pcr.boolean(1)
Mask= [pcr.ordinal(0)]*len(CRUTSSParamList)
for Param in CRUTSSParamList:
    -echoing parameter to screen
    print '\t\t%s' % Param
    -unzipping all data to directory
    PathOut= os.path.join(CRUTSSPath,Param)
    FileNameIn= os.path.join(CRUArchivePath,'cru_%s.zip' % (Param))
    try:
        os.mkdir(CRUTSSPath)
        os.mkdir(PathOut)
    except:
        pass
    zipToDir(FileNameIn,PathOut)
    -initializing bins
    SX= [pcr.scalar(0)]*NMonths
    SSQ= [pcr.scalar(0)]*NMonths
    N= [pcr.scalar(0)]*NMonths
    for YearOrd in range(StartYear,EndYear+1):
        for MonthOrd in range(1,NMonths+1):
            ID= pcr.ordinal(YearOrd*100+MonthOrd)
            X= pcr.readmap(os.path.join(PathOut,'c%s%04d.%03d' % \
                (Param,YearOrd,MonthOrd)))
            XValid=
pcr.ifthenelse((X+2730)>=0,pcr.boolean(1),pcr.boolean(0))
            XValid= pcr.cover(XValid,pcr.boolean(0))
            if YearOrd == StartYear and MonthOrd == 1:
                Mask[ParamCnt]= pcr.ifthenelse(XValid,ID,Mask[ParamCnt])
            else:
                Mask[ParamCnt]=
pcr.ifthenelse(pcr.pcrand(pcr.pcrnot(XValid),\
                (Mask[ParamCnt] > 0)),ID,Mask[ParamCnt])
            if YearOrd >= ClimStart and YearOrd <= ClimEnd:

```

```

        adding data
        SX[MonthOrd-1]= SX[MonthOrd-1]+X
        SSQ[MonthOrd-1]= SSQ[MonthOrd-1]+X**2
        N[MonthOrd-1]= N[MonthOrd-
1]+pcr.ifthenelse(XValid,pcr.scalar(1),0)
    for MonthOrd in range(1,NMonths+1):
        -calculating mean and standard deviation
        AVG= pcr.ifthen(N[MonthOrd-1]>=1,SX[MonthOrd-1]/N[MonthOrd-1])
        SD= pcr.ifthen(N[MonthOrd-1]>=1,\
            1/N[MonthOrd-1]*(N[MonthOrd-1]*SSQ[MonthOrd-1]-SX[MonthOrd-
1]**2)**0.5)
        pcr.report(AVG,os.path.join(CRUclimPath,'clm%sav.%03d' %
(Param,MonthOrd)))
        pcr.report(SD,os.path.join(CRUclimPath,'clm%ssd.%03d' %
(Param,MonthOrd)))
        -reporting mask and updating ParamCnt
        CRUTSSMask= pcr.ifthenelse(pcr.pcrand(Mask[ParamCnt] <> \
            pcr.ordinal(StartYear*100+1),\
            Mask[ParamCnt] > 0),pcr.boolean(0),CRUTSSMask)
        pcr.report(Mask[ParamCnt],os.path.join(CRUclimPath,'cru_ext%s.map' % Param))
        ParamCnt+= 1
    2) any change in landmask over years and between params
    pcr.report(CRUTSSMask,os.path.join(CRUclimPath,'cru_extent.map'))
    test,testValid= pcr.cellvalue(pcr.mapminimum(pcr.scalar(CRUTSSMask)),1)
    if test == 0:
        some error occurred, no further processing
        print '\t\tCRU timeseries is not contiguous over space and time'
    All done
    print '\t\tall CRU fields processed'
    3) create patch for climatology on the basis of holdridge classification and
nearest point
    -conditional processing if CRUTSSMask returns only true values
    print '\t%s' % 'creating patch for climatology of wind speed and radiation'
    print '\t%s' % 'on the basis of the nearest cell with same Holdridge
classification'
    -Holdridge classification
    PrecAnnual= pcr.scalar(0)
    TempAnnual= pcr.scalar(0)
    for MonthOrd in range(1,NMonths+1):
        PrecAnnual= PrecAnnual+pcr.readmap(os.path.join(CRUclimPath,'clm%sav.%03d' %
('pre',MonthOrd)))
        TempAnnual= TempAnnual+pcr.readmap(os.path.join(CRUclimPath,'clm%sav.%03d' %
('tmp',MonthOrd)))
    PrecAnnual= pcr.scalar(0.1)*PrecAnnual
    TempAnnual= pcr.scalar(0.1/12.0)*TempAnnual
    HolClass= pcr.lookupnominal('maps/holdridge_classification.tbl',\
        PrecAnnual,TempAnnual)
    pcr.report(PrecAnnual,os.path.join(CRUclimPath,'clm%stot.map' % 'pre'))
    pcr.report(TempAnnual,os.path.join(CRUclimPath,'clm%stot.map' % 'tmp'))
    pcr.report(HolClass,'maps/holdridge.map')
    HolClass= pcr.cover(HolClass,0)
    -area covered by climatology and ID map
    Param= 'rad'
    X= pcr.readmap(os.path.join(CRUclimPath,'c%s%04d.%03d' %\
        (Param,6190,1)))
    CRUclimMask= pcr.ifthenelse(X >= 0,pcr.boolean(1),pcr.boolean(0))
    Param= 'wnd'
    X= pcr.readmap(os.path.join(CRUclimPath,'c%s%04d.%03d' %\
        (Param,6190,1)))
    CRUclimMask= pcr.cover(pcr.ifthenelse(pcr.abs(X) >=
0,CRUclimMask,pcr.boolean(0)),0)
    CRUTSSMask= pcr.cover(pcr.ifthenelse(Mask[0]>0,CRUTSSMask,pcr.boolean(0)),0)
    CRUPatch= pcr.pcrxor(CRUclimMask,CRUTSSMask)
    CRUclimID= pcr.nominal(pcr.uniqueid(CRUclimMask))
    pcr.report(CRUclimMask,'maps/cruclimmask.map')
    pcr.report(CRUPatch,'maps/crupatch.map')
    CRUPatchID= pcr.nominal(0)
    for Class in range(1,57):

```

```

-stepping through holdridge zones
CRUclimIDSel= pcr.ifthenelse(HolClass == pcr.nominal(Class),\
    CRUclimID,0)
CRUclimIDSpread= pcr.spreadzone(CRUclimIDSel,pcr.scalar(0),pcr.scalar(1))
CRUPatchID= pcr.ifthenelse(pcr.pcrand(CRUPatch,HolClass ==
pcr.nominal(Class)),\
    CRUclimIDSpread,CRUPatchID)
CRUPatchID= pcr.ifthenelse(CRUPatch,CRUPatchID,CRUclimID)
pcr.report(CRUPatchID,'maps/crupatchid.map')
-patch constructed
print '\t\t%s' % 'patch constructed'
4) Calculate evapotranspiration
print '\t\t%s' % '\tcalculating reference evapotranspiration'
-creating new climatology
print '\t\t%s' % 'patching wind and radiation fields'
for MonthOrd in range(1,NMonths+1):
    for Param in ['rfr','wnd']:
        X= pcr.readmap(os.path.join(CRUclimPath,'c%s%04d.%03d' %\
            (Param,6190,MonthOrd)))
        Y= pcr.ifthenelse(CRUPatch,pcr.areaaverage(X,CRUPatchID),X)
        pcr.report(Y,os.path.join(CRUclimPath,'x%s%04d.%03d' %\
            (Param,6190,MonthOrd)))
-creating output directory
Param= 'etp'
PathOut= os.path.join(CRUTSSPath,Param)
FileNameIn= os.path.join(CRUArchivePath,'cru_%s.zip' % (Param))
try:
    os.mkdir(CRUTSSPath)
    os.mkdir(PathOut)
except:
    pass
YearCnt= 0
for YearOrd in range(StartYear,EndYear+1):
    YearCnt+=1
    print '\t\t%s%04d' %('processing year ',YearOrd)
    command= 'pcrcalc -d temp.map -f penmon_cru.txt %04d' % YearOrd
    os.system(command)
    for MonthOrd in range(1,13):
        FileNameIn= os.path.join(PathOut,'%s%04d.%03d' %
('epot',YearOrd,MonthOrd))
        FileNameOut= os.path.join(PathOut,'%s%04d.%03d' %
('cetp',YearOrd,MonthOrd))
        shutil.move(FileNameIn,FileNameOut)
print '\t\t%s' % 'all monthly evapotranspiration fields calculated'
5) deriving climatology for evaptranspiration
print '\t\t%s' % 'calculating climatology over 1961-1990 for:'
ParamCnt= 0
Mask= [pcr.ordinal(0)]*len(CRUTSSParamList)
for Param in ['etp']:
    -setting path
    PathOut= os.path.join(CRUTSSPath,Param)
    -echoing parameter to screen
    print '\t\t%s' % Param
    -initializing bins
    SX= [pcr.scalar(0)]*NMonths
    SSQ= [pcr.scalar(0)]*NMonths
    N= [pcr.scalar(0)]*NMonths
    for YearOrd in range(StartYear,EndYear+1):
        for MonthOrd in range(1,NMonths+1):
            ID= pcr.ordinal(YearOrd*100+MonthOrd)
            X= pcr.readmap(os.path.join(PathOut,'c%s%04d.%03d' %\
                (Param,YearOrd,MonthOrd)))
            XValid=
pcr.ifthenelse((X+2730)>=0,pcr.boolean(1),pcr.boolean(0))
            XValid= pcr.cover(XValid,pcr.boolean(0))
            if YearOrd == StartYear and MonthOrd == 1:
                Mask[ParamCnt]= pcr.ifthenelse(XValid,ID,Mask[ParamCnt])
            else:

```

```

Mask[ParamCnt]=
pcr.ifthenelse(pcr.pcrand(pcr.pcrnot(XValid),\
    (Mask[ParamCnt] > 0)),ID,Mask[ParamCnt])
    if YearOrd >= ClimStart and YearOrd <= ClimEnd:
        adding data
        SX[MonthOrd-1]= SX[MonthOrd-1]+X
        SSQ[MonthOrd-1]= SSQ[MonthOrd-1]+X**2
        N[MonthOrd-1]= N[MonthOrd-
1]+pcr.ifthenelse(XValid,pcr.scalar(1),0)
    for MonthOrd in range(1,NMonths+1):
        -calculating mean and standard deviation
        AVG= pcr.ifthen(N[MonthOrd-1]>=1,SX[MonthOrd-1]/N[MonthOrd-1])
        SD= pcr.ifthen(N[MonthOrd-1]>=1,\
            1/N[MonthOrd-1]*(N[MonthOrd-1]*SSQ[MonthOrd-1]-SX[MonthOrd-
1]**2)**0.5)
        pcr.report(AVG,os.path.join(CRUclimPath,'clm%sav.%03d' %
(Param,MonthOrd)))
        pcr.report(SD,os.path.join(CRUclimPath,'clm%ssd.%03d' %
(Param,MonthOrd)))
        -reporting mask and updating ParamCnt
        pcr.report(Mask[ParamCnt],os.path.join(CRUclimPath,'cru_ext%s.map' % Param))
        ParamCnt+= 1
    -total over climatology
EpotAnnual= pcr.scalar(0)
for MonthOrd in range(1,NMonths+1):
    EpotAnnual= EpotAnnual+pcr.readmap(os.path.join(CRUclimPath,'clm%sav.%03d' %
('etp',MonthOrd)))
pcr.report(pcr.scalar(365/12)*EpotAnnual,os.path.join(CRUclimPath,'clm%stot.map' %
'etp'))
#6) creating growing season
print '\t%s' % 'obtaining growing season and growth function'
#-create directory if not existing already
try:
    os.mkdir(CRUGrowthPath)
except:
    pass
#-processing temperature, precipitation and evaporation for growth season of
natural vegetation
#-crop factor variables
CropVegMax= 1.20
CropVegMin= 0.20
CropVegStep= 0.01
CropVeg= pcr.scalar(CropVegMax)
CropStack= [pcr.boolean(0.0)]*12
#-temperature limit curtailing growing season
tmpLimit= pcr.scalar(5.0)
#-meteo Stacks
tmpStack= [pcr.scalar(0.0)]*12
preStack= [pcr.scalar(0.0)]*12
preMax= pcr.scalar(0.0)
#-moisture values and stacks
moiststorMax= pcr.scalar(300.0)
moistgrowthLim= pcr.scalar(125.0)
moiststorHi= pcr.scalar(0.0)
moiststorLo= pcr.scalar(0.0)
moiststorStack= [pcr.scalar(0.0)]*12
moistdefStack= [pcr.scalar(0.0)]*12
#-stacks for functions specifying growth
fTStack= [pcr.scalar(0.0)]*12
fSStack= [pcr.scalar(0.0)]*12
fGStack= [pcr.scalar(0.0)]*12
#-patch for relative LAI, estimated crop factor and meteo factor
jointClone= pcr.pcror(CloneMap,cruClone)
#-points of nodes on growth curve and stack for months:
# 0) no particulars, 1) start of growth, 2) start of mid-season,
# 3) end of mid-season, 4) end of senescence
# and timing thereof
nodeStack= [pcr.scalar(0.0)]*12

```

```

stepStack= [pcr.scalar(0.0)]*12
devStack= [pcr.scalar(0.0)]*12
relLAIStack= [pcr.scalar(0.0)]*12
emergenceLim= pcr.scalar(0.10)
matureLim= pcr.scalar(0.80)
senescenceLim= pcr.scalar(0.60)
fGMaxOcc= pcr.scalar(-1)
emergenceStart= pcr.scalar(-1)
matureStart= pcr.scalar(-1)
senescenceStart= pcr.scalar(-1)
dormancyStart= pcr.scalar(-1)
#-obtaining BATS temperature function and
# determining start, end and length of growing season
#-reading in stacks of temperature and precipitation
tmpMin= tmpLimit
tmpMax= pcr.scalar(-1.0e2)
growthStart= pcr.scalar(-1)
growthEnd= pcr.scalar(-1)
growthLength= pcr.scalar(0)
growthMultiple= pcr.boolean(0)
for MonthOrd in range(1,13):
    tmpX=
pcr.scalar(CRUTSSConvList[0])*pcr.readmap(os.path.join(CRUclimPath, 'clm%sav.%03d' %
\
    ('tmp',MonthOrd)))
    tmpStack[MonthOrd-1]= tmpX
    tmpMax= pcr.ifthenelse(tmpX>=tmpLimit,pcr.max(tmpX,tmpMax),tmpMax)
    preX= pcr.scalar(CRUTSSConvList[1])*pcr.readmap(os.path.join(CRUclimPath,\
    'clm%sav.%03d' % ('pre',MonthOrd)))
    preStack[MonthOrd-1]= preX
    preMax= pcr.max(preX,preMax)
#-looping through years to process growing season temporal characteristics
# and obtaining BATS function
for MonthOrd in range(0,12):
    fTStack[MonthOrd]= pcr.ifthenelse(tmpStack[MonthOrd]>tmpLimit,\
    pcr.scalar(1.0)-pcr.min(1,(tmpMax-tmpStack[MonthOrd])/\
    (tmpMax-tmpMin))**2,0)
    growthLength=
growthLength+pcr.ifthenelse(tmpStack[MonthOrd]>=tmpLimit,pcr.scalar(1),\
    pcr.scalar(0))
    growthMultiple=
pcr.ifthenelse(pcr.pcrand(pcr.pcrand(tmpStack[MonthOrd]>=tmpLimit,\
    tmpStack[MonthOrd-1]<tmpLimit),growthStart>0),\
    pcr.boolean(1),growthMultiple)
    growthStart= pcr.ifthenelse(pcr.pcrand(tmpStack[MonthOrd]>=tmpLimit,\
    tmpStack[MonthOrd-1]<tmpLimit),pcr.scalar(MonthOrd+1),growthStart)
    growthEnd= pcr.ifthenelse(pcr.pcrand(tmpStack[MonthOrd]<tmpLimit,\
    tmpStack[MonthOrd-1]>=tmpLimit),pcr.scalar(MonthOrd+1),growthEnd)
#-water balance
#looped through CropVegList, updated if moistStorHi is zero
CropCntMax= int(CropVegMax/CropVegStep+0.5)
CropCntMin= int(CropVegMin/CropVegStep-0.5)
for CropCnt in range(CropCntMax,CropCntMin,-1):
    CropVegValue= CropVegMin+(CropCnt-(CropCntMin+1))*CropVegStep
    print '\t\t%s%4.2f' % ('crop factor: ',CropVegValue)
    CropVeg= pcr.ifthenelse(pcr.pcrand(moiststorHi>moistgrowthLim,\
    moiststorLo>pcr.scalar(-1.0)*moistgrowthLim),\
    CropVeg,pcr.scalar(CropVegValue))
    CropBare= min(CropVeg,pcr.scalar(0.2))
    #-obtaining crop factor
    for MonthOrd in range(0,12):
        CropStack[MonthOrd]= CropBare+0.5*(fTStack[MonthOrd-1]+\
        fTStack[MonthOrd])*(CropVeg-CropBare)
    #-resetting moiststorLo to maximum available storage
    moiststorLo= moiststorMax
    #-stepping through five years to get dynamic equilibrium for water balance,
    # moisture storage in mm and cumulative moisture deficit and
    for YearOrd in range(0,5):

```

```

    for MonthOrd in range(1,13):
        preX= preStack[MonthOrd-1]
        etpX= CropStack[MonthOrd-1]*pcr.scalar(\
            1000*(JulianDay[MonthOrd]-JulianDay[MonthOrd-1]))*\
            pcr.readmap(os.path.join(CRUCLimPath, 'clm%sav.%03d' %\
                ('etp',MonthOrd)))
        moiststorStack[MonthOrd-1]=
pcr.min(moiststorMax,moiststorStack[MonthOrd-2]+preX-etpX)
        moistdefStack[MonthOrd-1]=
pcr.ifthenelse(moiststorStack[MonthOrd-1]<0,\
            moistdefStack[MonthOrd-2]+moiststorStack[MonthOrd-1],0)
        moiststorStack[MonthOrd-1]= pcr.max(0,moiststorStack[MonthOrd-
1])

        moiststorHi= pcr.max(moiststorStack[MonthOrd-1],moiststorHi)
        moiststorLo= pcr.min(moistdefStack[MonthOrd-1],moiststorLo)
#-boolean map of areas without any positive storage to
# patch f(S) for these areas on the basis of 2-month average precipitation
desert= pcr.ifthenelse(moiststorHi>0.0,pcr.boolean(0),pcr.boolean(1))
#for MonthOrd in range(0,12):
#-obtaining reduction function due to water stress for areas with ephemeral water
shortage
# and growth function per month and peak growth
fGMax= pcr.scalar(-9.9)
for MonthOrd in range(0,12):
    weightFactor= 0
    for icnt in range(0,1):
        weightFactor+= 1
        Month= MonthOrd+icnt
        if Month>11:
            Month= Month-12
            fSStack[MonthOrd]= fSStack[MonthOrd]+preStack[Month]
            fSStack[MonthOrd]= 1.0/weightFactor*fSStack[MonthOrd]/preMax
            fSStack[MonthOrd]= pcr.ifthenelse(desert, fSStack[MonthOrd],\
                pcr.max(0,1+moistdefStack[MonthOrd]/moiststorHi))
            fGStack[MonthOrd]= fTStack[MonthOrd]*fSStack[MonthOrd]
            fGMax= pcr.max(fGMax,fGStack[MonthOrd])
#-patch for points that have temperature above threshold but have no growth
# (no moisture during growingseason) and scaling values between 0 and 1
laiPatch= pcr.ifthenelse(cruClone,pcr.pcrand(fGMax>emergenceLim,\
    tmpMax>tmpLimit),pcr.pcrnot(CloneMap))
CRUPatchSel= pcr.nominal(pcr.pcrand(cruClone,laiPatch))
CRUPatchID= pcr.spreadzone(CRUPatchSel,pcr.scalar(0),pcr.scalar(1))
fGMax= pcr.scalar(-9.9)
fGMin= pcr.scalar(9.9)
for MonthOrd in range(0,12):
    fGStack[MonthOrd]= pcr.ifthen(laiPatch,fGStack[MonthOrd])
    fGStack[MonthOrd]= pcr.ifthen(jointClone,pcr.cover(fGStack[MonthOrd],\
        pcr.areaaverage(fGStack[MonthOrd],CRUPatchID)))
    fGMin= pcr.min(fGMin,fGStack[MonthOrd])
    fGMax= pcr.max(fGMax,fGStack[MonthOrd])
for MonthOrd in range(0,12):
    fGMaxOcc= pcr.ifthenelse(fGStack[MonthOrd]==fGMax,pcr.scalar(MonthOrd+1),\
        fGMaxOcc)
    fGStack[MonthOrd]= (fGStack[MonthOrd]-fGMin)/(fGMax-fGMin)
    fGStack[MonthOrd]=
pcr.cover(pcr.ifthenelse(tmpMax>tmpLimit,fGStack[MonthOrd],0),\
        fGStack[MonthOrd])
    fGStack[MonthOrd]= pcr.min(matureLim,pcr.max(emergenceLim,fGStack[MonthOrd]))
#patching crop factor
CropVeg= pcr.ifthen(jointClone,pcr.cover(CropVeg,\
    pcr.areaaverage(CropVeg,CRUPatchID)))
#-determining nodes of growth functions
for MonthOrd in range(0,12):
    MonthPrev= MonthOrd-1
    MonthSucc= MonthOrd+1
    if MonthSucc>11:
        MonthSucc= MonthSucc-12
#-emergence

```

```

nodeStack[MonthOrd]=
pcr.ifthenelse(pcr.pcrand(fGStack[MonthOrd]<=emergenceLim,\
    fGStack[MonthSucc]>emergenceLim),pcr.scalar(1),nodeStack[MonthOrd])
#-start mid-season
nodeStack[MonthOrd]= pcr.ifthenelse(pcr.pcrand(fGStack[MonthOrd]>=
matureLim,\
    fGStack[MonthPrev]<matureLim),pcr.scalar(2),nodeStack[MonthOrd])
#-end mid-season
nodeStack[MonthOrd]= pcr.ifthenelse(pcr.pcrand(fGStack[MonthOrd]>=matureLim,\
    fGStack[MonthSucc]<matureLim),pcr.scalar(3),nodeStack[MonthOrd])
#-senescence
nodeStack[MonthOrd]=
pcr.ifthenelse(pcr.pcrand(fGStack[MonthOrd]>=senescenceLim,\
    fGStack[MonthSucc]<senescenceLim),pcr.scalar(4),nodeStack[MonthOrd])
#-selecting nodes of interest from node types
emergenceDuration= pcr.scalar(13)
matureDuration= pcr.scalar(13)
senescenceDuration= pcr.scalar(13)
dormancyDuration= pcr.scalar(13)
for MonthOrd in range(0,12):
    #-map of current month
    MonthMap= pcr.scalar(MonthOrd+1)
    #duration from month to peak of growth
    Duration= fGMaxOcc-
MonthMap+pcr.ifthenelse(MonthMap>fGMaxOcc,pcr.scalar(12),0)
    # 1) emergence
    emergenceCond=
pcr.ifthenelse(nodeStack[MonthOrd]==1,pcr.boolean(1),pcr.boolean(0))
    emergenceStart= pcr.ifthenelse(pcr.pcrand(Duration<emergenceDuration,\
    emergenceCond),MonthMap,emergenceStart)
    emergenceDuration= pcr.ifthenelse(emergenceCond,\
    pcr.min(Duration,emergenceDuration),emergenceDuration)
for MonthOrd in range(0,12):
    #-map of current month
    MonthMap= pcr.scalar(MonthOrd+1)
    #duration from month to peak of growth
    Duration= fGMaxOcc-
MonthMap+pcr.ifthenelse(MonthMap>fGMaxOcc,pcr.scalar(12),0)
    # 2) start of maturity
    matureCond=
pcr.ifthenelse(pcr.pcrand(nodeStack[MonthOrd]==2,emergenceDuration>Duration),\
    pcr.boolean(1),pcr.boolean(0))
    matureStart= pcr.ifthenelse(pcr.pcrand(Duration<matureDuration,\
    matureCond),MonthMap,matureStart)
    matureDuration= pcr.ifthenelse(matureCond,\
    pcr.min(Duration,matureDuration),matureDuration)
for MonthOrd in range(0,12):
    #-map of current month
    MonthMap= pcr.scalar(MonthOrd+1)
    #duration from peak of growth to month
    Duration= MonthMap-
fGMaxOcc+pcr.ifthenelse(MonthMap<fGMaxOcc,pcr.scalar(12),0)
    # 3) start of senescence
    senescenceCond= pcr.ifthenelse(pcr.pcrand(nodeStack[MonthOrd]==3,\
    pcr.pcrand(fGStack[MonthOrd]==matureLim,nodeStack[MonthOrd]==4)),\
    pcr.boolean(1),pcr.boolean(0))
    senescenceStart= pcr.ifthenelse(pcr.pcrand(Duration<senescenceDuration,\
    senescenceCond),MonthMap,senescenceStart)
    senescenceDuration= pcr.ifthenelse(senescenceCond,\
    pcr.min(Duration,senescenceDuration),senescenceDuration)
    # 4) start of dormancy, reset if regrowth occurs and value is closer
    #to senescence than to emergence
    dormancyReset= pcr.ifthenelse(pcr.pcrand(dormancyDuration<13,\
    pcr.pcrand(nodeStack[MonthOrd]==2,nodeStack[MonthOrd]==3)),\
    pcr.boolean(1),pcr.boolean(0))
    dormancyReset= pcr.ifthenelse(Duration<MonthMap-emergenceStart+\
    pcr.ifthenelse(MonthMap<emergenceStart,pcr.scalar(12),0),\
    dormancyReset,pcr.boolean(0))

```

```

    dormancyDuration=
pcr.ifthenelse(dormancyReset,pcr.scalar(13),dormancyDuration)
    dormancyCond=
pcr.ifthenelse(nodeStack[MonthOrd]==4,pcr.boolean(1),pcr.boolean(0))
    dormancyStart= pcr.ifthenelse(pcr.pcrand(Duration<dormancyDuration,\
    dormancyCond),MonthMap,dormancyStart)
    dormancyDuration= pcr.ifthenelse(dormancyCond,\
    pcr.min(Duration,dormancyDuration),dormancyDuration)
#-patching start of growth season for desert regions
matureStart= pcr.ifthenelse(matureStart<0,senescenceStart,matureStart)
#-obtaining step function and deviations thereof
# 1) emergence and full growth
#-timers and slope
MonthMaturity= matureStart-emergenceStart+\
    pcr.ifthenelse(matureStart<emergenceStart,pcr.scalar(12),0)
MonthSenescence= senescenceStart-emergenceStart+\
    pcr.ifthenelse(senescenceStart<emergenceStart,pcr.scalar(12),0)
growthSlope= (matureLim-emergenceLim)/MonthMaturity
#-interpolation between emergence and full growth conditions
for MonthOrd in range(0,12):
    MonthMap= pcr.scalar(MonthOrd+1)
    MonthMap= MonthMap-
emergenceStart+pcr.ifthenelse(MonthMap<emergenceStart,pcr.scalar(12),0)
    stepStack[MonthOrd]= pcr.ifthenelse(MonthMap<MonthMaturity,\
    MonthMap*growthSlope,0)+emergenceLim
    stepStack[MonthOrd]= pcr.ifthenelse(pcr.pcrand(MonthMap>=MonthMaturity,\
    MonthMap<=MonthSenescence),matureLim,stepStack[MonthOrd])
# 2) senescence and dormancy after full growth
#-timers and slope
MonthSenescence= dormancyStart-senescenceStart+\
    pcr.ifthenelse(dormancyStart<senescenceStart,pcr.scalar(12),0)
MonthEmergence= emergenceStart-senescenceStart+\
    pcr.ifthenelse(emergenceStart<senescenceStart,pcr.scalar(12),0)
senescenceSlope= pcr.scalar(0)
dormancySlope= pcr.scalar(0)
for MonthOrd in range(0,12):
    MonthPrev= MonthOrd-1
    MonthSucc= MonthOrd+1
    if MonthSucc>11:
        MonthSucc= MonthSucc-12
    dormancyCond= pcr.ifthenelse(pcr.scalar(MonthOrd+1)==dormancyStart,\
    pcr.boolean(1),pcr.boolean(0))
    senescenceSlope= pcr.ifthenelse(dormancyCond,\
    pcr.ifthenelse(MonthSenescence>0,\
    (fgStack[MonthOrd]-matureLim)/MonthSenescence,pcr.scalar(-
1)),senescenceSlope)
    dormancySlope= pcr.ifthenelse(dormancyCond,\
    pcr.min(fgStack[MonthOrd]-fgStack[MonthPrev],\
    fgStack[MonthSucc]-fgStack[MonthOrd]),dormancySlope)
    senescenceLim= pcr.ifthenelse(dormancyCond,fgStack[MonthOrd],senescenceLim)
#-interpolation between end of full growth and start of senescence
for MonthOrd in range(0,12):
    MonthMap= pcr.scalar(MonthOrd+1)
    MonthMap= MonthMap-
senescenceStart+pcr.ifthenelse(MonthMap<senescenceStart,pcr.scalar(12),0)
    stepStack[MonthOrd]= pcr.ifthenelse(MonthMap<MonthSenescence,\
    matureLim+MonthMap*senescenceSlope,stepStack[MonthOrd])
    stepStack[MonthOrd]= pcr.ifthenelse(pcr.pcrand(MonthMap>=MonthSenescence,\
    MonthMap<MonthEmergence),pcr.max(emergenceLim,senescenceLim+\
    (MonthMap-MonthSenescence)*dormancySlope),stepStack[MonthOrd])
#-scaling growth functions between 0 and 1 and obtaining deviations
# to scale relative LAI
for MonthOrd in range(0,12):
    fgStack[MonthOrd]= (fgStack[MonthOrd]-emergenceLim)/(matureLim-emergenceLim)
    stepStack[MonthOrd]= (stepStack[MonthOrd]-emergenceLim)/(matureLim-
emergenceLim)
    devStack[MonthOrd]= pcr.scalar(1.0)-0.5*((CropVeg-CropVegMin)/(CropVegMax-
CropVegMin)+\

```

```

    fgStack[MonthOrd]-stepStack[MonthOrd])
    devStack[MonthOrd]= pcr.ifthenelse(stepStack[MonthOrd]>0,\
    stepStack[MonthOrd]**devStack[MonthOrd],stepStack[MonthOrd])
#-check for any erroneous spikes
for MonthOrd in range(0,12):
    MonthPrev= MonthOrd-1
    MonthSucc= MonthOrd+1
    if MonthSucc>11:
        MonthSucc= MonthSucc-12
    slopePrev= devStack[MonthPrev]-devStack[MonthOrd]
    slopeSucc= devStack[MonthOrd]-devStack[MonthSucc]
    slopePrev= pcr.ifthenelse(slopePrev==0,pcr.scalar(0),\
    slopePrev/pcr.abs(slopePrev))
    slopeSucc= pcr.ifthenelse(slopeSucc==0,pcr.scalar(0),\
    slopeSucc/pcr.abs(slopeSucc))
    changeCond= pcr.ifthenelse(pcr.pcror(slopePrev==0,slopeSucc==0),\
    pcr.boolean(0),pcr.ifthenelse(slopePrev==slopeSucc,pcr.boolean(0),\
    pcr.boolean(1)))
    changeCond= pcr.ifthenelse(pcr.pcrand(devStack[MonthOrd]>emergenceLim,\
    devStack[MonthOrd]<matureLim),changeCond,pcr.boolean(0))
    relLAIStack[MonthOrd]= pcr.ifthenelse(changeCond,\
    0.5*(devStack[MonthPrev]+devStack[MonthSucc]),\
    devStack[MonthOrd])
#-calculating growth length characteristics and check on generated output
fullGrowthSeason= pcr.scalar(0)
optGrowthSeason= pcr.scalar(0)
laiRelCover= pcr.boolean(1)
for MonthOrd in range(0,12):
    optGrowthSeason= optGrowthSeason+\
    pcr.ifthenelse(relLAIStack[MonthOrd]==1,pcr.scalar(1),pcr.scalar(0))
    fullGrowthSeason= fullGrowthSeason+\
    pcr.ifthenelse(relLAIStack[MonthOrd]>0,pcr.scalar(1),pcr.scalar(0))
    laiRelCover= pcr.ifthenelse(cruClone,\
    pcr.ifthenelse(pcr.cover(relLAIStack[MonthOrd],-1)>=
0,laiRelCover,pcr.boolean(0)),\
    pcr.boolean(1))
growthSeasonAnomaly=
pcr.ifthen(cruClone,pcr.ifthenelse(optGrowthSeason>fullGrowthSeason,\
    pcr.boolean(1),0))
laiMinAnomaly= pcr.boolean(0)
laiMaxAnomaly= pcr.boolean(0)
for MonthOrd in range(0,12):
    MonthMap= pcr.scalar(MonthOrd+1)
    laiMinAnomaly= pcr.ifthenelse(MonthMap==
emergenceStart,pcr.ifthenelse(relLAIStack[MonthOrd]<>\
    0.0,pcr.boolean(1),pcr.boolean(0)),laiMinAnomaly)
    laiMaxAnomaly= pcr.ifthenelse(MonthMap==
fgMaxOcc,pcr.ifthenelse(relLAIStack[MonthOrd]<> 1.0,\
    pcr.boolean(1),pcr.boolean(0)),laiMaxAnomaly)
    laiMinAnomaly= pcr.cover(laiMinAnomaly,0)
    laiMaxAnomaly= pcr.cover(laiMaxAnomaly,0)
#-calculation of meteorological factor for crop factor
# for month at which growth function is at its maximum
MetFactor= pcr.scalar(0.0)
for MonthOrd in range(1,13):
    wndX= pcr.scalar(0.75)*pcr.scalar(1.0)*\
    pcr.readmap(os.path.join(CRUCLimPath,'x%s%04d.%03d' %
('wnd',6190,MonthOrd)))
    tmpX= pcr.scalar(0.1)*\
    pcr.readmap(os.path.join(CRUCLimPath,'clm%sav.%03d' % ('tmp',MonthOrd)))
    deltaTmp= pcr.scalar(0.1)*\
    pcr.readmap(os.path.join(CRUCLimPath,'clm%sav.%03d' % ('dtr',MonthOrd)))
    maxTmp= tmpX+0.5*deltaTmp
    minTmp= tmpX-0.5*deltaTmp-\
    pcr.ifthenelse(moistdefStack >=0,pcr.scalar(0.0),pcr.scalar(2.0))
    RHMin=
pcr.scalar(100.0)*pcr.scalar(611.0)*pcr.exp(17.27*minTmp/(minTmp+237.3))/\
    (pcr.scalar(611.0)*pcr.exp(17.27*maxTmp/(maxTmp+237.3)))

```

```

    RHMin= pcr.min(80.0,pcr.max(RHMin,20))
    wndX= pcr.max(1.0,pcr.min(wndX,6.0));
    MetFactor= pcr.ifthenelse(fGMaxOcc==pcr.scalar(MonthOrd),\
        pcr.scalar(0.04)*(wndX-pcr.scalar(2.0))-\  

        pcr.scalar(0.004)*(RHMin-45),MetFactor)
#-patching meteo factor
MetFactor=
pcr.ifthenelse(jointClone,pcr.cover(MetFactor,pcr.areaaverage(MetFactor,CRUPatchID)
),0)
#-reporting maps of interest and exporting ASCII maps
pcr.report(growthLength,os.path.join(CRUGrowthPath,'growthlength.map'))
pcr.report(growthStart,os.path.join(CRUGrowthPath,'growthstart.map'))
pcr.report(growthEnd,os.path.join(CRUGrowthPath,'growthend.map'))
pcr.report(growthMultiple,os.path.join(CRUGrowthPath,'growthmultiple.map'))
pcr.report(moiststorHi,os.path.join(CRUGrowthPath,'moistmax.map'))
pcr.report(moiststorLo,os.path.join(CRUGrowthPath,'moistmin.map'))
pcr.report(tmpMin,os.path.join(CRUClimPath,'clmtmpmin.map'))
pcr.report(tmpMax,os.path.join(CRUClimPath,'clmtmpmax.map'))
pcr.report(growthSeasonAnomaly,os.path.join(CRUGrowthPath,'growthanomaly.map'))
pcr.report(laiMinAnomaly,os.path.join(CRUGrowthPath,'laimin_ano.map'))
pcr.report(laiMaxAnomaly,os.path.join(CRUGrowthPath,'laimax_ano.map'))
pcr.report(laiRelCover,os.path.join(CRUGrowthPath,'lai_mv.map'))
pcr.report(laiPatch,os.path.join(CRUGrowthPath,'laipatch.map'))
pcr.report(CropVeg,os.path.join(CRUGrowthPath,'cropveg.map'))
pcr.report(fGMaxOcc,os.path.join(CRUGrowthPath,'fg_max.map'))
pcr.report(emergenceStart,os.path.join(CRUGrowthPath,'emergence.map'))
pcr.report(matureStart,os.path.join(CRUGrowthPath,'maturity.map'))
pcr.report(senescenceStart,os.path.join(CRUGrowthPath,'senescence.map'))
pcr.report(dormancyStart,os.path.join(CRUGrowthPath,'dormancy.map'))
pcr.report(MetFactor,os.path.join(CRUGrowthPath,'metfactor.map'))
pcr.report(fullGrowthSeason,os.path.join(CRUGrowthPath,'fullseason.map'))
pcr.report(optGrowthSeason,os.path.join(CRUGrowthPath,'optseason.map'))
for MonthOrd in range(1,13):
    mapName= generateNameT(os.path.join(CRUGrowthPath,'tfunc'),MonthOrd)
    pcr.report(ftStack[MonthOrd-1],mapName)
    mapName= generateNameT(os.path.join(CRUGrowthPath,'sfunc'),MonthOrd)
    pcr.report(fsStack[MonthOrd-1],mapName)
    mapName= generateNameT(os.path.join(CRUGrowthPath,'gfunc'),MonthOrd)
    pcr.report(fgStack[MonthOrd-1],mapName)
    mapName= generateNameT(os.path.join(CRUGrowthPath,'moist'),MonthOrd)
    pcr.report(moiststorStack[MonthOrd-1],mapName)
    mapName= generateNameT(os.path.join(CRUGrowthPath,'moistdef'),MonthOrd)
    pcr.report(moistdefStack[MonthOrd-1],mapName)
    mapName= generateNameT(os.path.join(CRUGrowthPath,'node'),MonthOrd)
    pcr.report(nodeStack[MonthOrd-1],mapName)
    mapName= generateNameT(os.path.join(CRUGrowthPath,'step'),MonthOrd)
    pcr.report(stepStack[MonthOrd-1],mapName)
    mapName= generateNameT(os.path.join(CRUGrowthPath,'dev'),MonthOrd)
    pcr.report(devStack[MonthOrd-1],mapName)
    mapName= generateNameT(os.path.join(CRUGrowthPath,'lairel'),MonthOrd)
    pcr.report(rellAIStack[MonthOrd-1],mapName)
#-growth season calculated
print '\t\t%s' % 'distribution of growth season calculated'
#7) generating mean and sd of annual values for primary variables: P, T and EO
print '\t\t%s' % 'obtaining annual values and statistics'
#-create directory if not existing already
try:
    os.mkdir(CRUPeriod)
except:
    pass
#update parameter list
CRUTSSParamList[2]= 'etp'
CRUTSSConvList[2]= 1000
icnt= 0
for Param in CRUTSSParamList[0:3]:
    PathOut= os.path.join(CRUTSSPath,Param)
    convMap= pcr.scalar(CRUTSSConvList[icnt])
    icnt+=1

```

```

YearCnt= 0
for YearOrd in range(1901,2003):
    YearCnt+=1
    XAnnual= pcr.scalar(0)
    TotDays= pcr.scalar(0)
    for MonthOrd in range(1,13):
        NrDays= pcr.scalar(JulianDay[MonthOrd]-JulianDay[MonthOrd-1])
        if (calendar.isleap(YearOrd)and(MonthOrd == 2)):
            NrDays= NrDays+1
        TotDays= TotDays+NrDays
        X= convMap*pcr.readmap(os.path.join(PathOut,'c%s%04d.%03d' %\
            (Param,YearOrd,MonthOrd)))
        if (Param == 'tmp')or(Param=='etp'):
            X= NrDays*X
            XAnnual= XAnnual+X
    if Param == 'tmp':
        XAnnual= XAnnual/TotDays
    pcr.report(XAnnual,generateNameT(os.path.join(PathOut,'c%syearch' %\
        Param),YearCnt))
#-calculating statistics
command= 'pcrcalc -f cru_stat.txt %s c%syearch' % (Param,Param)
os.system(command)
for Period in ['', '1', '2', '3']:
    for Stat in ['avg', 'sd', 't']:
        FileNameIn= os.path.join(CRUPeriodPath, '%s%s.map' %
(Stat,Period))
        FileNameOut= os.path.join(CRUPeriodPath, '%s%s%s.map'
%(Param,Stat,Period))
        try:
            shutil.move(FileNameIn,FileNameOut)
        except:
            pass
        FileNameIn= os.path.join(CRUPeriodPath, '%s.tss' % ('avg'))
        FileNameOut= os.path.join(CRUPeriodPath, '%s.tss' %(Param))
        shutil.move(FileNameIn,FileNameOut)
#-all annual statistics calculated
print '\t\t%s' % 'annual statistics calculated'

```

Appendix 3: PCRaster script penmon_cru.txt

```
#estimating potential ETP with the Penman Equation
#for the CRU05 data set of average daily values per month
#on the basis of the CRU TS2.1 timeseries fields
#complemented by the CRU CL 1.0 at 0.5 degree resolution
#include updated calculation of short and longwave radiation on the basis of the
FAO (Allen et al.,1998)

binding
### INPUT ###
# global data
Mask = maps\cru_clone.map;           # clone map
SamplePoints= maps\globalsample.map;  # sample locations
DEM= maps\globaldem.map;             # dem

# table with fraction of relative sunshine hours given cloud cover
SUNFRACTBL= maps\sunhoursfrac.tbl;

# meteo from CRU05 dataset
# - time series
CLDStack= newlygridded\cld\ccld$1;   # Cloud cover in fraction and
conversion factor
CLDConv= scalar(0.001);
TMPStack= newlygridded\tmp\ctmp$1;   # Air temperature in degree C and
conversion factor
TMPConv= scalar(0.1);
VAPStack= newlygridded\vap\cvap$1;   # Vapour pressure in Pa and
conversion factor
VAPConv= scalar(10);
# - climatology over the period 1961-1990
WNDStack= climatology\xwnd6190;      # Windspeed in m/s and conversion factor,
patched to CRU TSS extent
WNDConv= scalar(1);
# - monthly climatology of extraterrestrial radiation
MAXRADStack= climatology\maxrad;     # maximum extraterrestrial radiation in
W/m2 and conversion factor
MAXRADConv= scalar(1);

### OUPUT ###
RAD= newlygridded\etp\rad;           #radiation terms: global radiation,
shortwave, longwave and net radiation
RSW= newlygridded\etp\rsw;
RLW= newlygridded\etp\rlw;
Er= newlygridded\etp\rnet;
EPOT= newlygridded\etp\epot$1;      # reference evapotranspiration ET0
EPOTTSS= newlygridded\etp\epot.tss;  # reported timeseries with potential
evaporation (mm/day)
EPOTANN= newlygridded\etp\epotannual.map; # total annual reference
evapotranspiration

### PENMAN CONSTANTS ###
Sigma= 5.67E-8;                      # Stefan-Boltzmann constant for emission of
longwave radiaton
Albedo= 0.23;                         # albedo for reference crop (grass)
Rs= 70.0;                              # surface resistance for reference
crop (grass)
Cp = 1004;                             # specific heat of air at constant P
(J/(kg*K))
Epsilon = 0.622;                      # ratio molecular weight of water vapour
and dry air
K = 0.41;                              # Karman constant
Roa = 1.2047;                          # air density (kg/m^3)
Windheight = 10.0;                    # height open area windspeed (m above
ground)
TempHeight= 2.0;                      # height open area temperature measurements
(m above ground)
Vegheight = 0.12;                     # height of the vegetation (m above ground)
```

```

### FAO CONSTANTS ###
RADCON= scalar(0.25);           #slope and constant to calculate incoming
shortwave radiation
RADSLOPE= scalar(0.50);        #on basis of sunshine fraction
RADDIF= scalar(0.35);          #basic reduction of relative incoming
radiation in correction of net longwave radiation
EA0= scalar(0.34);             #constant of correction of net longwave
radiation for vapour pressure
EAFACTOR= scalar(4.43e-3);     #corresponding factor

areamap
Mask;

timer
1 12 1;
repl= endtime;

initial
### Evapotranspiration ###

# Constant for longwave radiation correction for cloudiness
RADCOR= (1+RADDIF)/(RADCON+RADSLOPE);

# Constant in the windspeed part of aerodynamic resistance
# zero plane displacement, roughness height for momentum and heat and vapour
transfer
Zd= 2/3*Vegheight;
Z0m= 0.123*Vegheight;
Z0h= 0.1*Z0m;
RaTerm= ln((Windheight-Zd)/(Z0m))*ln((TempHeight-Zd)/(Z0h))/sqr(K);

# pressure, not measured: set to 1013 hPa at sea level
Press = 101300*((293-0.0065*cover(DEM,0))/293)**5.26;
# annual evapotranspiration initialized
EPOTANN= scalar(0);

dynamic

### PENMAN EVAPOTRANSPIRATION ###
# reading in timeseries: TSS
TMP= TMPConv*timeinput(TMPStack);           # daily temperature (i;½C)
CLD= CLDConv*timeinput(CLDStack);           # cloud cover (-)
Ea= VAPConv*timeinput(VAPStack);           # actual vapour pressure (Pa)
# reading in timeseries: climatology
WND = WNDConv*timeinput(WNDStack);         # windspeed (m/s)
WND= max(0.1,WND);
MAXRAD= MAXRADConv*timeinput(MAXRADStack); #maximum extraterrestrial
radiation (W/m2)

## Radiation term ##
# - sunshine fraction and corresponding fraction shortwave radiation
CLD1= roundoff(10*CLD+0.5);
CLD0= CLD1-1;
SUN0= lookupscalar(SUNFRACTBL,CLD0);
DELTASUN= (lookupscalar(SUNFRACTBL,CLD1)-SUN0)/(CLD1-CLD0);
SUNFRAC= SUN0+(10*CLD-CLD0)*DELTASUN;
RADFRAC= RADCON+RADSLOPE*SUNFRAC;
# - short, longwave and net radiation
RAD= RADFRAC*MAXRAD;
RSW= (1-Albedo)*RAD;
RLW= Sigma*(TMP+273)**4*max(0,EA0-
EAFACTOR*sqr(Ea))*(min(1+RADDIF,RADCOR*RADFRAC)-RADDIF);
Er= max(0,RSW-RLW);           # radiation (W/m2)

## Mass term ##
#Temperature relationships

```

```

Lv = 2.501E6-2370*TMP;                # latent heat of vaporization of
water (J/kg)
T237 = TMP+237.3;                    # term for Goff-relationship
Es = 611.0*exp(17.27*TMP/T237);      # saturated and actual vap pressure (Pa)
RH= Ea/Es;                            # not used

## wind term ##
# canopy resistance (s/m)
Ra = RaTerm/WND;

## weighing of radiation and mass transfer term ##
Delta = (4098.0*Es)/(T237**2);        # slope of sat vap pressure curve
(Pa/oC)
Gamma = (Cp*Press)/(Epsilon*Lv);      # psychrometric constant (Pa/oC)
DGLv = (Delta+Gamma*(1+Rs/Ra))*Lv;     # Pa/°C * J/kg

# Aerodynamic evaporation rate (mm/s)
Ea = Roa*Cp*(Es-Ea)/Ra;

## Potential evapotranspiration ##
Prad = Delta*Er/DGLv;
Pwind = Ea/DGLv;

#ETP in mm/s => m/day
report EPOT = max(0,0.001*(Prad+Pwind)*24*3600);
report EPOTTSS= timeoutoutput(SamplePoints,EPOT);
report (repl) EPOTANN= EPOTANN+EPOT*365/12;

```

Appendix 4: Subdivision of the Olson classification into the surface vegetation types used in PCR-GLOBWB (short, tall) and additional crop categories (natural vegetation, rain fed and irrigated crops). Only the ecosystem types occurring in the GLCC are shown. The values between brackets refer to the LAI during the growing season, LAI_g [$m^2 \cdot m^{-2}$], its range, $LAI_g - LAI_d$, and the estimated vegetation height, h [m], respectively.

Surface type	Crop category		
	Natural vegetation	Rain fed crops	Irrigated crops
Short vegetation	2: Low Sparse Grassland (1.8, 1.6, 0.3) 7: Tall Grasses and Shrubs (2.2, 1.4, 1.0) 8: Bare Desert (0.0, 0.0, 0.1) 9: Upland Tundra (2.3, 1.9, 0.3) 11: Semi Desert (0.5, 0.5, 0.1) 12: Glacier Ice (0.0, 0.0, 0.1) 13: Wooded Wet Swamp (3.5, 0.5, 0.3) 40: Cool Grasses and Shrubs (1.9, 1.9, 0.6) 41: Hot and Mild Grasses and Shrubs (1.7, 1.2, 1.0) 42: Cold Grassland (1.5, 0.6, 0.3) 44: Mire, Bog, Fen (2.5, 2.4, 0.3) 45: Marsh Wetland (2.5, 2.4, 0.3) 50: Sand Desert (0.0, 0.0, 0.1) 51: Semi Desert Shrubs (0.8, 0.3, 0.1) 52: Semi Desert Sage (1.2, 1.2, 0.1) 53: Barren Tundra (1.9, 1.9, 0.3) 64: Heath Scrub (4.6, 4.5, 1.0) 69: Polar and Alpine Desert (0.0, 0.0, 0.1)	1: Urban (0.0, 0.0, 10.0) 30: Cool Crops and Towns (2.5, 0.7, 1.0) 31: Crops and Town (4.5, 3.4, 1.0) 35: Corn and Beans Cropland (2.5, 2.1, 1.0) 93: Grass Crops (2.0, 2.0, 0.7) 94: Crops, Grass, Shrubs (2.7, 1.9, 1.0)	10: Irrigated Grassland (4.5, 4.5, 0.3) 36: Rice Paddy and Field (4.6, 4.6, 0.6) 37: Hot Irrigated Cropland (4.4, 3.1, 0.5) 38: Cool Irrigated Cropland (3.0, 3.0, 0.5) 76: Crop and Water Mixtures (4.4, 4.2, 0.6)
Tall vegetation	3: Coniferous Forest (9.2, 0.2, 10.0) 4: Deciduous Conifer Forest (3.6, 3.5, 10.0) 5: Deciduous Broadleaf Forest (5.2, 5.1, 10.0) 6: Evergreen Broadleaf Forests (9.9, 0.4, 6.8) 16: Shrub Evergreen (6.0, 3.8, 5.5) 17: Shrub Deciduous (4.6, 4.2, 2.6) 19: Evergreen Forest and Fields (6.0, 3.0, 2.5) 20: Cool Rain Forest (9.3, 0.0, 10.0) 21: Conifer Boreal Forest (5.5, 0.0, 10.0) 22: Cool Conifer Forest (9.2, 4.8, 10.0) 23: Cool Mixed Forest (4.2, 4.1, 10.0) 24: Mixed Forest (7.0, 6.0, 6.8) 25: Cool Broadleaf Forest (5.2, 4.7, 10.0) 26: Deciduous Broadleaf Forest (5.2, 4.4, 10.0) 27: Conifer Forest (9.7, 5.3, 10.0) 28: Montane Tropical Forests (4.8, 0.8, 5.5) 29: Seasonal Tropical Forest (9.1, 6.4, 10.0) 32: Dry Tropical Woods (6.1, 3.6, 5.5) 33: Tropical Rainforest (9.3, 0.0, 10.0) 43: Savanna (Woods) (3.0, 2.1, 2.5) 46: Mediterranean Scrub (4.3, 1.8, 4.6) 47: Dry Woody Scrub (4.6, 3.8, 2.6) 48: Dry Evergreen Woods (1.8, 0.1, 0.4)	34: Tropical Degraded Forest (6.0, 1.6, 5.5) 92: Broadleaf Crops (5.0, 2.7, 1.8) 95: Evergreen Tree Crop (6.0, 3.0, 2.5) 96: Deciduous Tree Crop (5.1, 3.6, 5.9)	

54: Cool Southern Hemisphere Mixed Forests (4.8, 4.7, 6.5)
55: Cool Fields and Woods (3.0, 2.1, 1.0)
56: Forest and Field (6.1, 3.9, 1.7)
57: Cool Forest and Field (4.0, 3.0, 1.7)
58: Fields and Woody Savanna (5.1, 3.0, 1.0)
59: Succulent and Thorn Scrub (4.7, 3.8, 0.3)
60: Small Leaf Mixed Woods (3.7, 3.6, 10.0)
61: Deciduous and Mixed Boreal Forest (4.7, 4.6, 6.5)
62: Narrow Conifers (3.4, 3.3, 3.1)
63: Wooded Tundra (3.1, 2.6, 0.5)
72: Mangrove (9.0, 0.0, 10.0)
78: Southern Hemisphere Mixed Forest (4.7, 4.6, 6.5)
79: Wet Sclerophytic Forest (4.8, 2.0, 6.5)
89: Moist Eucalyptus (4.8, 2.0, 6.5)
90: Rain Green Tropical Forest (9.3, 0.0, 10.0)
91: Woody Savanna (1.9, 0.8, 2.5)
