Delivery report

Development and validation of a global GPP/NPP model using MERIS and Sentinel-3 data (TerrA-P)

ATBD v2

Iain Colin Prentice, Rebecca Thomas

Study accomplished under the authority of ESA

March 2018

DISTRIBUTION LIST

Philippe Goryl, ESA-ESRIN

Iain Colin Prentice, ICL
Rebecca Thomas, ICL

Else Swinnen, VITO
Roel Van Hoolst, VITO
Bruno Smets, VITO

Ivan janssens, UA
Sara Vicca, UA
Manuela Balzarolo, UA

SUMMARY

The TerrA-P project proposes to implement and validate a new global monitoring system for primary production by land ecosystems, comprising dekad composites of gross primary production (GPP) and annual composites of above-ground biomass production (ABP) by C_3 and C_4 plants. Spectral reflectance data provided by Sentinel-3 will be used to drive a recently developed universal, first-principles light use efficiency model (the 'P' model) for GPP.

This document is version 2 of the Algorithm Theoretical Basis Document (ATBD) for the proposed GPP and ABP products. It describes the design criteria adopted by TerrA-P for both products based on a user survey, and also on a set of scientific requirements that go significantly beyond the current state of the art. It includes a brief review of the history of light use efficiency (LUE) models, and summarizes the strengths and weaknesses of various existing LUE models including those used operationally. It is shown how the P model – with the linear mathematical form of a LUE model – can be derived from the standard (non-linear) model of photosynthesis via explicit hypotheses for the environmentally induced acclimation of three parameters of the standard model. In comparisons with GPP derived from eddy-covariance carbon dioxide (CO_2) flux measurements, the P model has been shown to achieve comparable accuracy to other LUE models, while requiring fewer parameters to be estimated.

Just one parameter, the intrinsic quantum efficiency of C_3 photosynthesis $\phi_0(C_3)$, was calibrated to optimize the agreement of modelled and observed GPP across 17 eddy-covariance carbon dioxide (CO_2) flux measurement sites. The sites were selected for their relatively homogeneous surrounding vegetation and long measurement records. The present document updates the calibration presented in version 1 of the ATBD. In the new calibration, remotely sensed land surface temperature (LST) is used as an alternative to air temperature, to provide both temperature and vapour pressure deficit drivers for the model. The new calibration uses the same merged input data set as in version 1, based on spectral reflectances from SeaWiFS and MERIS, to provide the fraction of incident photosynthetically active radiation absorbed by green tissues (fAPAR). Meteorological data other than temperature were derived from site measurements as before. The new optimized value of $\phi_0(C_3)$ was 0.092, slightly higher than the value of 0.084 obtained in version 1, but still within the range expected based on leaf-level measurements.

The model has also been applied to data from a larger set of flux measurement sites for validation. Model outputs were supplied based on the new calibration, but driven by new and recently available fAPAR data for these sites. Model outputs for validation include a comprehensive uncertainty propagation scheme, summarized here, which provides time- and location-specific uncertainties for modelled GPP. This document also outlines the development strategy for the ABP product, and summarizes the validation and benchmarking strategies for both products.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

3-PG	Physiological Principles Predicting Growth
AATSR	Advanced Along-Track Scanning Radiometer
ABP	Above-ground biomass production
ATBD	Algorithm Theoretical Basis Document
AVHRR	Advanced Very High Resolution Radiometer
BESS	Breathing Earth System Simulator
BIOME-BGC	Biome–BioGeochemical Cycles model
BPLUT	Biome Parameter Look-Up Table
CASA	Carnegie-Ames-Stanford Approach model
C-Fix	Carbon Fixation model
CGLOPS-1	Copernicus Global Land Operations Lot1
CRO	Cropland
CUE	Carbon use efficiency
DBF	Deciduous broadleaf forest
DMP	Dry Matter Productivity algorithm
EBF	Evergreen broadleaf forest
EC-LUE	Eddy Covariance-Light Use Efficiency model
ECMWF	European Centre for Medium-range Weather Forecasts
ENF	Evergreen needleleaf forest
ET	Evapotranspiration
FACE	Free Air Carbon dioxide Enrichment
fAPAR, FPAR	Fractional absorbed photosynthetically active radiation
FvCB	Farquhar, von Caemmerer and Berry model
GLO-PEM	Global Production Efficiency Model
GMAO	Global Modeling and Assimilation Office
GPP	Gross primary production
GVI	Global vegetation index
IPAR	Incident photosynthetically active radiation
LAI	Leaf area index
LPJ	Lund-Potsdam-Jena model
LPX	Land-surface Processes and eXchanges model
LST	Land surface temperature
LUE	Light use efficiency
MERIS	Medium Resolution Imaging Spectrometer
MODIS	Moderate Resolution Imaging Spectroradiometer
MTCI	MERIS Total Chlorophyll Index
NASA	National Aeronautics and Space Administration
NDVI	Normalized difference vegetation index
NOAA	National Oceanic and Atmospheric Administration
NPP	Net primary production
OSH	Open shrubland
P	Production model
PPFD	Photosynthetic photon flux density
RMSE	Root mean squared error of prediction
SDBM	Simple Diagnostic Biosphere Model
SeaWiFS	Sea-Viewing Wide Field-of-View Sensor
VOC	Volatile organic compound
vpd	vapour pressure deficit

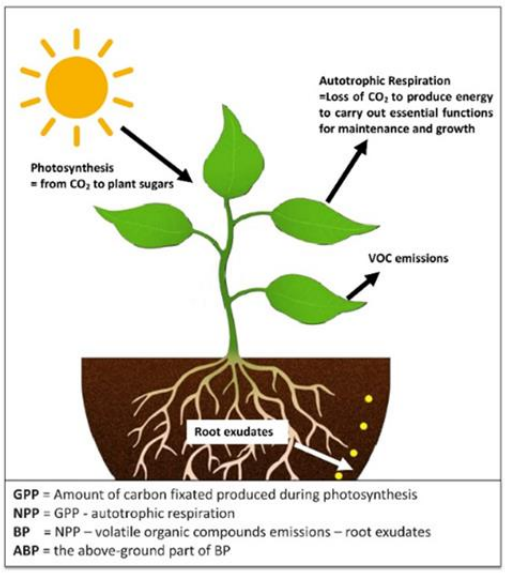
LIST OF SYMBOLS

C_3	photosynthetic pathway whereby carbon is first incorporated into three-carbon sugars
C_4	photosynthetic pathway whereby carbon is first incorporated into four-carbon sugars
c_a	ambient partial pressure of carbon dioxide [Pa]
c_i	leaf-internal partial pressure of carbon dioxide [Pa]
C:N	ratio of carbon to nitrogen content [–]
CO_2	carbon dioxide
c^*	unit cost of maintaining electron transport capacity (P model) [–]
D	vapour pressure deficit [kPa]
e_a	absolute vapour pressure of water [Pa]
e_s	saturation vapour pressure of water [Pa]
f_{BG}	fraction of carbon allocated below ground [–]
g_1	stomatal sensitivity parameter (Medlyn et al., 2011 model) [$kPa^{0.5}$]
J_{max}	maximum rate of electron transport (FvCB model) [µmol m^{-2} s^{-1}]
m	intermediate term in the P model [–]
K	effective Michaelis-Menten coefficient of Rubisco [Pa]
K_C	Michaelis-Menten coefficient of Rubisco for carboxylation [Pa]
K_O	Michaelis-Menten coefficient of Rubisco for oxygenation [Pa]
O	partial pressure of oxygen [Pa]
pH	measure of acidity: minus log (hydrogen ion concentration) [–]
Q_{10}	ratio of reaction rate at temperature T + 10 K to reaction rate at temperature T [–]
R	universal gas constant [J mol^{-1} K^{-1}]
R^2	coefficient of determination [–]
T	absolute temperature [K]
T_C	Celsius temperature [˚C]
u^2	standard uncertainty
V_{cmax}	maximum velocity of carboxylation (FvCB model) [µmol m^{-2} s^{-1}]
β	ratio of cost factors for carboxylation and transpiration (P model) [–]
Γ*	CO_2 compensation point in the absence of mitochondrial respiration (FvCB model) [Pa]
$\delta^{13}C$	normalized ratio of the stable carbon isotopes ^{13}C and ^{12}C relative to a standard reference [‰]
$\Delta H_{\Gamma*}$	activation energy of Γ* [kJ mol^{-1}]
ΔH_{K_C}	activation energy of K_C [kJ mol^{-1}]
ΔH_{K_O}	activation energy of K_O [kJ mol^{-1}]
η*	viscosity of water relative to its value at 25˚C [–]
Θ	curvature parameter in the relation between electron transport and absorbed PPFD (FvCB model) [–]
ξ	stomatal sensitivity parameter (P model) [$kPa^{0.5}$]
$\phi_0(C_3)$	intrinsic quantum efficiency of C_3 photosynthesis [–]
$\phi_0(C_4)$	intrinsic quantum efficiency of C_4 photosynthesis [–]
χ	ratio of leaf-internal to ambient carbon dioxide [–]
$\partial f(x_i)/\partial x_i$	sensitivity of GPP to variable x_i

CHAPTER 1 BACKGROUND OF THE DOCUMENT

1.1. SCOPE AND OBJECTIVES

The TerrA-P project aims to implement and validate a new global monitoring system for primary production by land ecosystems. The project focuses on gross primary production (GPP) and above-ground biomass production (ABP), defined as follows:



Gross primary production is the rate of total carbon fixation (photosynthesis) by the ecosystem. This is the most fundamental measure of primary production, as all other ecosystem functions depend on it. Also, thanks to the availability of eddy-covariance flux measurements, GPP data are available – at time scales from half-hourly up to multi-annual – for some hundreds of locations worldwide (albeit with a bias towards temperate regions), and for croplands as well as for natural and managed ecosystems.

Figure 1 (left): schematic illustrating the different aspects of primary production.

Above-ground biomass production is the rate of production of plant matter, excluding roots. This is a practically important measure, because this is the production rate of forage for grazing animals; it is closely related to the production rate of timber for harvest; and it can be converted (through data on the harvest index – the ratio of harvestable yield to ABP – for different crops) to estimates of crop yield. There are data on ABP, occasionally at a monthly time scale but more commonly at the annual time scale, for many ecosystems, especially crops and managed forests but also for natural ecosystems.

We chose not to focus either on (total) biomass production or on net primary production (NPP), for the following reasons:

Biomass production is the total rate of production of plant matter, including roots. For most crops the root production is of less interest than the above-ground production. Even for root crops there are data on the harvest index, so ABP can be used to predict harvestable yield by a simple conversion. There are some data on BP but in most cases the root production has not been measured directly, but rather inferred from above-ground measurements. Inevitably this increases the uncertainty of BP data.

Net primary production is defined as the difference between GPP and plant respiration. Formerly, NPP was assumed to equivalent to BP. Most data that claim to be NPP are, in fact, BP. But it is now

understood that a fraction of NPP – under some circumstances this can be as much as 20% – is "lost" from the plant by other pathways than respiration, in the form of volatile organic compounds (such as isoprene and monoterpenes) and/or root exudates, which do not contribute to biomass production. Moreover, most published data sets of "NPP" are of poor quality.

The aims of the project will be achieved by using spectral reflectance data from Sentinel-3 as input to a recently developed universal, first-principles light use efficiency (LUE) model for GPP, called the 'P' model (for 'production'). Initial calibration was carried out (and reported in ATBD v1) against GPP data derived from eddy-covariance carbon dioxide (CO_2) flux measurements at 17 selected sites. This calibration used a merged data set from SeaWiFS and MERIS (Global Vegetation Index, GVI) to provide the fraction of incident photosynthetically active radiation that is absorbed by green tissues (fAPAR), a key input to the P model. The model also requires meteorological data, all of which were derived for the initial calibration from direct measurements at the flux sites. The updated calibration presented here is based on the same fAPAR data, but remotely sensed land surface temperature (LST) from the Advanced Along-Track Scanning Radiometer (AASTR) has been used to provide the temperature and vapour pressure deficit drivers for modelled photosynthesis. This substitution is desirable theoretically, because LST is expected to be closer to the actual temperature of leaves; and practically for the global application, because remotely sensed LST is retrieved at a higher spatial resolution than can realistically be achieved by interpolation of weather-station or meteorological analysis data.

The project will create a prototype processing chain for 10-daily GPP and annual ABP by C_3 and C_4 plants. It will include data quality flags and a specification of per-pixel uncertainties, which has been implemented in the creation of model outputs for validation. Validation of the model is described in a separate document led by the University of Antwerp group. Validation has been based on a more extensive data set of GPP, derived from eddy-covariance flux measurements in different biomes and climatic regions. ABP predictions when available will be validated by comparison to the most comprehensive available global set of quality-controlled data on annual ABP, which has been compiled by the University of Antwerp group.

1.2. CONTENT OF THE DOCUMENT

The document is organized in the following way.

- **Chapter 1** describes the background of the document.
- **Chapter 2** describes the criteria adopted for new primary production data products, taking into account both the requirements articulated by users, and scientific considerations.
- **Chapter 3** is a selective review of existing approaches to monitoring primary production from space, including those currently used operationally.
- **Chapter 4** describes the P model and the proposed manner of its implementation, including the approach used to calculate uncertainties; introduces the GPP calibration data set; and presents the results of the new calibration.
- **Chapter 5** summarizes the approach to validation and benchmarking. The detailed validation of modelled GPP is the subject of a separate document.

CHAPTER 2 CRITERIA FOR NEW PRIMARY PRODUCTION DATA PRODUCTS

2.1. INTERPRETATION OF USER REQUIREMENTS

TerrA-P has conducted a survey targeting various potential users of the proposed new products, including current users of the Copernicus Global Land Operations Lot1 (CGLOPS-1) of the Copernicus Global Land Service. The full details of the survey are described in the Requirements Baseline Document. It yielded the following key information for product design.

The *principal products currently in use* are MODIS GPP and net primary production (NPP), CGLOPS-1 dry matter production (DMP) products, and models. The new products should aim to reproduce (at least) the functionality of these existing, widely-used products.

Different users work at different *geographic scales*, from subnational to global. The new products should accordingly be global, gridded products allowing flexibility of application.

About three-quarters of users surveyed agreed with the proposed strategy to *focus on GPP* and *ABP*. Some caveats were mentioned, including the fact that GPP cannot be directly derived from flux measurements (ecosystem respiration has to be factored out through a 'partitioning' method, of which there are several that give somewhat different results); and the fact that total biomass production (including production below ground) may sometimes be of greater interest than ABP. We propose to deal with uncertainty in partitioning by using the range of alternative partitioning methods as a measure of uncertainty in observed GPP. For biomass production, however, the paucity and low reliability of data on below-ground production argues for maintaining a focus on ABP. We note that ABP is quantitatively related to below-ground production by, for example, root crops just as it is quantitatively related to above-ground production by grain crops.

Users were approximately equally divided in their preferences for *units of dry matter versus carbon*. For maximum comparability with existing products, and with the main data sources for each quantity, we propose supplying GPP in carbon units and ABP in dry matter units. Climate modellers preferred carbon units, but are likely to be more interested in 10-daily GPP than annual ABP.

Most users saw the need to consider *C_3 versus C_4 photosynthesis* but, not surprisingly, there was no specific proposal as to how the prevalence of the two pathways could be specified on a per-pixel basis. We propose to circumvent this problem by providing both as alternatives for every pixel.

Most users asked for a *data quality layer*, and *information on uncertainty*. A per-pixel uncertainty layer was not explicitly requested. However, a numbers of users in different ways indicated a need for quantitative, per-pixel uncertainty information. In our view a systematic approach to per-pixel uncertainty should be a significant part of product development, and would satisfy this need.

The most popular *sampling frequency* was 10-daily. A number of users voted for daily, but daily data on spectral reflectances are not meaningful because many dates, in most locations, will be affected by clouds. This problem can be largely circumvented by providing 10-daily composites. Most users asked for *data availability* in near-real time, that is, 3 to 5 days after acquisition.

The preferred *spatial resolution* is 300 m. Some users work with coarser resolutions which, however, can be readily obtained by post-processing of a 300 m product.

The average request in terms of *relative accuracy* was close to 20%, which likely provides a realistic target accuracy for both 10-daily GPP and annual ABP.

Table 1 summarizes the user requirements, and the specifications adopted that are consistent with these requirements as far as is technically feasible.

Table 1: Summary of user requirements and specifications adopted for TerrA-P products.

User requirement	Specification adopted
Geographic scale subnational to global	Global gridded product
Focus on GPP and ABP	Focus on GBP and ABP
Carbon or dry-matter units	Carbon for GPP, dry matter for ABP
Distinction of C_3 and C_4 photosynthesis	Provide results for both C_3 and C_4 plants
Data quality layer, information on uncertainty	Provide per-pixel uncertainty estimates
Daily to 10-daily sampling frequency	10-daily sampling frequency
Data availability in near-real time	Data available 3-5 days after acquisition
Spatial resolution 300 m or coarser	300 m grid with facility for post-processing
Relative accuracy ca 20%	Target relative accuracy 20%

2.2. SCIENTIFIC REQUIREMENTS

A priori we determined that new products should as far as possible possible satisfy a number of additional criteria, summarized here. These requirements go significantly beyond the current state of the art in LUE-based modelling.

Explicit relationship to the standard model of photosynthesis. The Farquhar, von Caemmerer and Berry (1980) (FvCB) model is the standard model of C_3 photosynthesis, and modifications exist to describe C_4 photosynthesis. There are thousands of published field measurements of the parameters defined in the FcVB model. All current ecophysiological theory, and the great majority of biophysical land-surface schemes for climate modelling, make use of the FvCB model. Therefore, a newly developed remotely sensed GPP product should be explicitly defined in terms of the FvCB model.

There is no such general model for plant respiration and other carbon "losses" from GPP. Thus models for biomass production should be based on GPP, with modifications to account for these losses as fractions of GPP.

Representation of physiological effects of CO_2. Models based on remotely sensed data, including those in operational use, generally do not consider the effect of changing atmospheric CO_2 concentration on the LUE of photosynthesis. Thus, they only consider a CO_2 effect in so far as it is manifested by changes in foliage cover that can be seen from space. As a direct consequence, products such as MODIS GPP and NPP severely underestimate the generally increasing trend in primary production due to rising CO_2 (De Kauwe et al., 2016a). Newly developed products should explicitly include the effect of CO_2 concentration on LUE.

No discontinuities at biome boundaries. Although the convention of defining different model parameter values per biome is entrenched in remote sensing applications, it inevitably leads to discontinuities at boundaries defined by an external classification. This is a highly undesirable property, because biomes intergrade. Imposed biome boundaries are arbitrary and differently located according to different land cover products. New products should attempt to avoid such discontinuities.

A demonstrated level of accuracy assessed by comparison to relevant measurements. Eddy-covariance measurements of CO_2 flux can be processed ('partitioned') to yield estimates of daily, 10-daily, monthly or annual GPP. Flux measurement sites vary in public availability status, and in the length of records. Some sites are more suitable than others for model calibration and validation, because in areas of complex terrain or land use patterns there can be a severe problem in attempting to match remotely sensed spectral reflectance data with the (time-varying) footprint of the flux tower. Thus, model calibration and validation should be based on an informed selection of flux measurement sites.

CHAPTER 3 REVIEW OF SELECTED EXISTING APPROACHES

3.1. BACKGROUND AND HISTORY

The basis for nearly all algorithms designed to calculate primary production based on remotely sensed data is the general LUE model first proposed by Monteith (1972, 1977). Monteith based this model on field measurements of crop growth in both tropical and temperate climates, showing that growth is proportional to the time-integral of the light absorbed by the crop.

The general LUE model can be applied either to GPP or to NPP. More formally, in a remote sensing context, the general LUE model states that primary production is proportional to the product of incident photosynthetic photon flux density (PPFD) and fractional green vegetation cover, also called fractional absorbed photosynthetically active radiation (fAPAR or FPAR). fAPAR depends on Leaf Area Index (LAI) but is closer to actual reflectance measurements than LAI, and more directly related to primary production. In the remote sensing literature, incident PPFD (μmol m^{-2} s^{-1}) is more often described as 'incident photosynthetically active radiation' (IPAR) (W m^{-2}). The former term is more accurate because photosynthesis depends on the *number* of photons absorbed, rather than their energy (which varies with their wavelength). However, the two units can be interconverted, if it is assumed that the solar spectrum is constant.

NPP is the remainder of GPP after autotrophic (plant) respiration has converted approximately half of GPP back to CO_2. Traditionally, NPP has been regarded as synonymous with biomass production, i.e. the production of plant tissues. However, it is now recognized that a fraction (which can be as much as 20%) of NPP is lost from plants in the form of exudation from roots (a carbon subsidy to microbes in the rhizosphere, which enables plants to increase their uptake of soil nutrients) and emissions of volatile organic compounds (VOCs) such as isoprene and monoterpenes from leaves (which confer protection against both oxidants, including ozone, and high leaf temperatures). We therefore make a distinction between NPP and biomass production. The latter is of greater interest than NPP *sensu stricto* to users in forestry and agriculture.

Pioneering examples of remotely sensed primary production models are the Simple Diagnostic Biosphere Model (SDBM) of Knorr and Heimann (1995), the Carnegie-Ames-Stanford Approach (CASA) model of Potter et al. (1993), and the Global Production Efficiency Model (GLO-PEM) of Prince and Goward (1995). These models used spectral reflectance data from the Advanced Very High Resolution Radiometer (AVHRR) to infer fAPAR. A constant maximum LUE was specified, then reduced by scalars representing aspects of temperature and moisture conditions that can reduce LUE. The SDBM was combined with an atmospheric tracer transport model and deployed in an inverse mode, using observations of the seasonal cycle of atmospheric CO_2 concentration at different latitudes to estimate a single global maximum value for the LUE of NPP, and a single global Q_{10} value to quantify the dependence of soil organic matter decomposition on temperature. In GLO-PEM, theoretical maximum LUE values for GPP were determined based on the FvCB model. One value was assigned for C_3 plants and another for C_4 plants. These values were modified following the FvCB model's estimation of photorespiratory carbon loss as a function of temperature. GLO-PEM also made use of a number of other remote-sensing approaches to estimate meteorological variables, including IPAR. Unusually, this modelling approach – further

developed by Goetz et al. (1999) – aimed to derive all required meteorological and biophysical variables from remotely sensed platforms.

The Physiological Principles Predicting Growth (3-PG) model of Landsberg and Waring (1997) is also an LUE model and can be driven by remotely sensed and meteorological data. This model, focused on forest production, makes a number of empirically well-supported simplifications to predict GPP, NPP and forest growth.

All of the models discussed so far included an estimate of soil moisture availability as one of the factors reducing LUE.

In the following subsections, we review the principal literature on LUE models and note the key features of various widely used models, including those used operationally, and some other models that have introduced potentially useful innovations.

3.2. THE MODIS GPP AND NPP PRODUCTS

By far the most widely used remotely sensed primary production data products for scientific applications today are the MODIS GPP and NPP (MOD17) products (Running et al., 2004; Zhao et al., 2005). The most recent (2015) user's guide to the MOD17 products can be found at: http://www.ntsg.umt.edu/sites/ntsg.umt.edu/files/modis/MOD17UsersGuide2015_v3.pdf
MODIS GPP specifies a maximum LUE which varies per biome. This value is multiplied by two scalars formulated as ramp functions: a linearly declining function of daily vapour pressure deficit (vpd), between biome-specific limit values where the function equals one or zero, and a linearly increasing function of daily minimum temperature, between biome-specific limit values where the function equals zero or one. Soil moisture effects are not considered: thus implicitly, soil moisture effects are considered to be accounted for in the remotely sensed fAPAR, and/or the effect of vpd on photosynthesis. The calculations make use of a land-cover classification and a Biome Parameter Look-Up Table (BPLUT) which recognizes 10 biomes. The required meteorological data, including IPAR (incoming solar shortwave radiation multiplied by 0.45), are supplied by the Global Modeling and Assimilation Office (GMAO) of the US National Aeronautics and Space Administration (NASA) at a 1˚ x 1.25˚ grid resolution. These data are smoothly interpolated to the finer (1 km) spatial resolution of the remotely sensed fAPAR data. The spatial and temporal resolution (8 days) of the product are set by the fAPAR data, which are obtained from the MODIS FPAR/LAI product.

Biome-specific model parameter values for MODIS GPP were based on literature-derived estimates used in the process-based BIOME-Biogeochemical Cycles model (BIOME-BGC: Running and Hunt, 1993). Thus, CO_2 flux data were not used in the initial calibration of the remote-sensing based model. However, the MODIS GPP product has been very extensively and independently evaluated by comparison with GPP derived from flux measurements. A number of these evaluation studies are cited in the user guide. Verma et al. (2014) included MODIS GPP – and also a version called MOD17-Tower, which was pre-calibrated against flux measurements – in a systematic global comparison with flux data-derived GPP. MODIS GPP was included in the set of seven LUE models compared globally with flux data-derived GPP by Yuan et al. (2014). Tang et al. (2015) undertook a comprehensive evaluation of MODIS GPP against flux data-derived GPP for forest ecosystems. Tagesson et al. (2017) showed that MODIS GPP greatly underestimates flux data-derived GPP in the Sahel, apparently due to unrealistically low maximum LUE assigned to semi-arid ecosystems.

The approach taken in the MODIS NPP product to derive NPP from GPP relies on separately modelling autotrophic respiration, which is separated into maintenance and growth components.

NPP is then the difference between modelled GPP and modelled total autotrophic respiration. Maintenance respiration in the model depends on leaf area index (LAI), obtained from the MODIS FPAR/LAI product, and biome-specific values of specific leaf area, the ratio of fine root mass to leaf mass, base maintenance respiration rates for leaves and fine roots, and a Q_{10} value that specifies the steepness of an assumed exponential relationship to temperature. In the current version of MODIS NPP, the Q_{10} of leaf maintenance respiration is a decreasing function of growth temperature. This is said to reflect respiratory acclimation, although Medlyn (2011) pointed out that the use of this function in fact further steepens the modelled negative response of NPP to temperature. Other components of maintenance respiration assume a fixed Q_{10}. Additional biome-specific information is used to infer annual growth respiration, which is then combined with annually integrated estimates of GPP and autotrophic respiration to yield annual NPP. Evaluation of NPP is considerably more difficult than evaluation of GPP, as the available data are far more limited. A number of compilations of annual NPP measurements have been made, however, and these have been used as a benchmark for MODIS NPP.

The MODIS GPP and NPP products were the first global and widely disseminated products of their kind. They represented a major technical advance, and now underpin a large number of high-profile scientific publications. However, criticisms have been made of some recent studies in which key assumptions underlying these products were overlooked. The study by Zhao and Running (2010) for example was criticized by Samanta et al. (2011) and Medlyn (2011), as the decline in NPP during the 2000 to 2009 period reported by Zhao and Running (2010) was not present in the remotely sensed data. It was, instead, a consequence of the high sensitivity of maintenance respiration to temperature in the model. Prentice (2013) noted that this sensitivity must be too high, because the interannual variability of MODIS NPP is so large as to fully account for the observed year-to-year variability of the atmospheric CO_2 growth rate – allowing no room for the effect of temperature variability on the soil decomposition rate, which is generally understood to be the key factor modulating the CO_2 growth rate (e.g. Wenzel et al., 2014).

Another emerging problem is the weak increase over time shown in the MODIS GPP and NPP products. This weak trend contrasts with the strong increases shown by most process-based models (Anav et al., 2015), the attribution of increasing measured GPP at flux sites to rising CO_2 (Fernández-Martínez et al., 2017), and the evidence for increasing LUE as the principal driver of a large amplification of the high-latitude seasonal cycle of atmospheric CO_2 (Graven et al., 2013; Wenzel et al., 2016; Thomas et al., 2017).

In process-based models, GPP and NPP increase primarily as a consequence of the rising atmospheric CO_2 concentration. Smith et al. (2016) asserted that process-based models overestimate the stimulatory effect of rising CO_2 concentrations on NPP. They reached this conclusion by comparing process-based model outputs with the much weaker trend shown by MODIS NPP. However, MODIS GPP and NPP do not allow CO_2 concentration to influence LUE; therefore, the only possible CO_2 influence in these models is via increasing fAPAR. There is a worldwide 'greening' trend which has been attributed in part to the effect of CO_2 in increasing NPP and water use efficiency (Donohue et al., 2013; Ukkola et al., 2015; Zhu et al., 2016). But Free Air Carbon Enrichment (FACE) experiments have shown that the principal effect of enhanced CO_2 on primary production in forests is through increased LUE, whereas increased LAI or fAPAR are much less important. Thus, the discrepancy noted by Smith et al. (2016) does not mean that process-based models overestimate the effect of CO_2 on NPP. Instead it means that MODIS NPP underestimates this effect (De Kauwe et al., 2016a), as a consequence of its design.

3.3. C-FIX AND THE DRY MATTER PRODUCTIVITY PRODUCT

Veroustraete et al. (1994, 2002) introduced the C-Fix model, a pioneer effort and the forerunner of the present-day Dry Matter Productivity (DMP) algorithm (Swinnen et al., 2015) currently provided by CGLOPS1. The original version of C-Fix was a LUE model for GPP in which a maximum LUE (a single global value) was reduced through multiplication by scalars representing the effects of suboptimal temperature and water availability. Values for NPP were obtained from GPP using an empirical linear function of temperature to account for the fraction of GPP lost to autotrophic respiration.

C-Fix also included an attempt to account for an effect of rising atmospheric CO_2 concentration on photosynthesis via the FvCB model. However, the effect was implemented by way of the CO_2-dependence of the Rubisco-limited rate of photosynthesis, which is nearly always the rate that is measured when leaves are subjected to saturating light intensity (De Kauwe et al., 2016b). This rate depends steeply on CO_2. But it is not the rate actually realized in the field, at lower average light intensity. Under typical daytime field conditions the light- and Rubisco-limited rates are approximately equal (Maire et al., 2012). The light-limited rate depends on CO_2 as well, but less steeply than the Rubisco-limited rate. The original CO_2 response function in C-Fix therefore presumably overestimates the effect of CO_2 on GPP. However this overestimation may be tempered by the fact that no distinction is made between leaf-internal and ambient CO_2. The steepness of the response is thereby less than it would have been if leaf-internal CO_2 had been used.

Another limitation of C-Fix is its assumption of a universal, strongly peaked response of GPP to temperature with an optimum around 22˚C – thus not accounting for thermal acclimation, and necessarily underestimating GPP in hot climates. But C-Fix was designed for use in temperate forests and has in fact only been applied in temperate regions. Veroustraete et al. (2002) successfully compared C-Fix GPP predictions with flux data from forests in different regions of Europe.

C-Fix was not deployed operationally, but it provided the initial basis for the present operational DMP product (Swinnen et al., 2015). In DMP, IPAR is determined from solar shortwave radiation by applying a factor 0.48, and APAR is obtained by multiplying IPAR by a remotely sensed estimate of fAPAR, as in other models. Daily meteorological data supplied by the European Centre for Medium range Weather Forecasts (ECMWF) on a 0.25˚ grid are bilinearly interpolated to the remote-sensing pixels. A single global maximum value is assigned to the maximum LUE. This is modified by a temperature function and (for NPP) a further temperature function. These two functions are unchanged from C-Fix. Production is therefore likely to be underestimated in hot climates, due to the ‘temperate’ location of the peak of the temperature response function for LUE. There is no effect of water availability (apart from that manifested in changes in fAPAR) and the CO_2 response of C-Fix is not implemented. C_4 photosynthesis is not distinguished.

3.4. SOME RECENT DEVELOPMENTS AND TRENDS

Numerous LUE models have been developed in recent years with the expressed intention of achieving improved consistency with CO_2 flux-based measurements of GPP. These measurements, when suitably analysed, can provide more information about the controls of LUE than is utilized in an ‘end-of-pipe’ comparison of modelled and measured values of GPP.

The EC-LUE model (Yuan et al., 2007) requires only four quantities as input: fAPAR (which is estimated from the Normalized Difference Vegetation Index, NDVI), IPAR, air temperature and the Bowen ratio, which was inferred from other remotely sensed measurements (Yuan et al., 2007). In a later version the Bowen ratio was replaced by the ratio of actual evapotranspiration to net radiation (Yuan et al., 2010), which proved to be more robustly estimated than the Bowen ratio. EC-LUE has the merit of simplicity, as well as outperforming MODIS GPP in comparisons with flux measurements.

Further innovations in recent LUE models include consideration of seasonal variations in maximum LUE (Garbulsky et al., 2010; Lin et al., 2017), accounting for thermal acclimation (McCallum et al., 2013), the inclusion of an effect of diffuse light fraction on LUE (Donohue et al., 2014), and the use of a MODIS canopy conductance product to include effects of vpd and soil moisture availability (Yebra et al., 2015).

These various recent model developments differ in the extent to which they achieve good empirical results by increasing complexity and adding to the number of unknown parameters, *versus* the alternative of trying to make modelling more robust through the application of theory that can often simplify models and reduce the number of unknown parameters. McCallum et al. (2009), for example, argued for the inclusion of all of those processes that have been shown to improve model performance. An opposite view (Prentice et al., 2015) is that this is a flawed approach that tends always to increase uncertainty, rather than to reduce it, as the number of parameters increases and the transparency of the model decreases.

3.5. THE SCARF MODEL

Ogutu et al. (2013) introduced the Southampton Carbon Flux (SCARF) model, a new LUE model with a number of specific advantages for potential operational use. First, the remote sensing data used to drive the model are the MERIS Total Chlorophyll Index (MTCI). MTCI was considered by Ogutu et al. (2013) to offer an improvement over more standard 'greenness' measures as it explicitly relates to the abundance of green, photosynthesizing tissues. They argued that other measures of fAPAR used in LUE models include light absorption by non-green tissues that do not contribute to GPP. Second, the model substantially avoids spatial discontinuities and the use of a look-up table for biome-specific parameters by (a) adopting universal intrinsic quantum efficiency values defined in terms of the FvCB model for C_3 and C_4 plants respectively, and (b) applying universal temperature and CO_2 response functions for C_3 plants, and vpd response functions for C_3 and C_4 plants. The CO_2 and temperature response functions for C_3 plants were derived from the FcVB model, but an empirical formulation was used for the vpd response functions. A look-up table (together with a number of simplifying assumptions) was used to estimate the fraction of C_3 versus C_4 photosynthesis on a per-pixel basis. The model was evaluated successfully against GPP data derived from flux measurements across Europe and the USA.

Ogutu and Dash (2013) showed that the fidelity of flux measurements to the FvCB model at two study sites was close enough that reasonable estimates of 'green' fAPAR could be obtained by inversion of the model, i.e. estimating the fAPAR required to produce the observed patterns of GPP. This finding strongly supports the notion that LUE models could avoid the need for multiple unknown parameters (including the need for look-up tables for vegetation types, apart from the issue of C_3 versus C_4 photosynthesis) through application of the FvCB model.

3.6. THE BESS MODEL

The Breathing Earth System Simulator (BESS) by Ryu et al. (2011) represents an advanced modelling system for GPP and evapotranspiration (ET) and included many novel features. The model was designed to be the 'first system that harmonizes and utilizes MODIS Atmosphere and Land products on the same projection and spatial resolution over the global land' (Ryu et al., 2011, p. 1), thereby utilizing remotely sensed solar radiation and other meteorological data (from MODIS) and avoiding the need to interpolate such data from a coarse spatial grid. The model was described as calibration-free, that is, no parameters were to be estimated from flux data; all were to be specified independently. A global selection of 33 flux measurement sites was used to independently evaluate the model's predictions of both GPP and ET.

BESS is more complex than any of the other models discussed in this review. It includes an explicit radiative transfer model for solar radiation in the canopy; a 'two-leaf' model that distinguishes the properties of sun and shade leaves, which has been claimed to provide better accuracy, especially in modelling the differential penetration of diffuse versus direct light into the canopy and the consequences for photosynthesis; consideration of foliar clumping effects on photosynthetic light absorption (making use of a satellite-derived foliar clumping index product); and an extended FvCB model, including light-, Rubisco- and triose phosphate utilization-limited rates of photosynthesis. However, this complexity comes at a considerable cost, both computational, and in terms of data availability.

The model was set up on the Microsoft Azure cloud computing system, as it was considered to be infeasible on the supercomputing resources available at Berkeley. Many compromises were unavoidably made. A look-up table was used to provide values of many parameters, including the carboxylation capacity (V_{cmax}) over much of the Earth's surface. For some biomes V_{cmax} was estimated from foliage N, which in turn was estimated from vegetation albedo – this approach relying on the (questionable) relationship between foliar N and $V_{cmax.}$ The ratio of leaf-internal to ambient CO_2 was set at constant values for C_3 and C_4 plants respectively, thus disregarding the well-established effect of vpd on this ratio. Some external data, not available from MODIS, were obtained from a coarsely gridded re-analysis product. Thus, although BESS includes many advanced features and the ideal of obtaining all required information from remote sensing remains worth pursuing, this ideal was not in fact realized. This approach does not appear to provide a useful way forward for the development of operational systems at the present time.

CHAPTER 4 THE P MODEL: DESCRIPTION OF THE PROPOSED METHOD

4.1. THE P MODEL

The P model is fully derived and described by Wang et al. (2017). Aspects of the underlying theory have been applied by Keenan et al. (2016) and Wang et al. (2016). Unlike any of the LUE models discussed above, the P model possesses all of the following desirable attributes for a 'next-generation' primary production monitoring system:

- An explicit derivation from the FvCB model, and a clear relationship to a well-established functional form for stomatal behaviour – both elements required for a prediction of GPP.
- A representation of physiological CO_2 effects on photosynthesis that is consistent with both the FvCB model and results from FACE experiments.
- No distinctions among plant functional types and biomes (except for the difference between C_3 and C_4 plants), eliminating the need for spatial discontinuities induced by the use of a land-cover classification and look-up table.
- Demonstrated success in representing flux-derived GPP across different biomes at monthly time scales.

The model is extremely parameter-sparse, while achieving a fidelity to data comparable with or better than other models. This combination of simplicity with accuracy has been achieved through the development of theory that accounts for the observed environmental dependencies of the ratio (henceforth termed χ) of the leaf-internal (c_i) to ambient (c_a) partial pressures of CO_2 in C_3 plants; and the acclimation of photosynthetic parameters in space and time. Both aspects of the theory rely on eco-evolutionary optimality concepts to derive testable hypotheses, which in turn yield good agreement with observations from field measurements and field experiments.

4.1.1. PREDICTING χ WITH THE LEAST-COST HYPOTHESIS

Prentice et al. (2014) tested a quantitative theory based on a hypothesis first proposed by Wright et al. (2004), that plants should minimize the sum of the unit costs (per unit of carbon assimilation) of maintaining the capacities for carboxylation (proportional to V_{cmax}) and water transport (proportional to the maximum rate of transpiration). Transpiration is a requirement for photosynthesis because stomata have to open to allow CO_2 to diffuse towards the chloroplasts. In so doing, they draw water from the soil to replenish that lost by evaporation through the stomata. Prentice et al. (2014) showed that this 'least-cost' criterion leads to an optimal value of χ as a function of environmental variables (temperature and vpd) that is independent of PAR and almost independent of c_a. This value is given by:

$$\chi = \Gamma^*/c_a + (1 - \Gamma^*/c_a)\,\xi/(\xi + \sqrt{D}), \quad (1a)$$

$$\xi = \sqrt{\{\beta(K + \Gamma^*)/1.6\eta^*\}} \quad (1b)$$

where Γ^* and K are respectively the photorespiratory compensation point and the effective Michaelis-Menten coefficient of Rubisco (both known functions of temperature and atmospheric

pressure), β is an empirical constant (estimated from $\delta^{13}C$ data: Wang et al., 2017), η* is the viscosity of water relative to its value at 25˚C (a known function of temperature), and D is the vpd. As $\Gamma^* << c_a$ under field conditions, and $K >> \Gamma^*$, equation (1) is well approximated by:

$$\chi = \xi/(\xi + \sqrt{D}) \quad (2a)$$

$$\xi = \sqrt{\{\beta K/1.6\eta^*\}} \quad (2b)$$

Equation (2a) is mathematically identical with the 'universal stomatal model' proposed by Medlyn et al. (2011) and tested against a globally distributed set of gas-exchange measurements by Lin et al. (2012). Lin et al. (2012) also showed that ξ (there called g_1) increases approximately exponentially with temperature. This was predicted by Medlyn et al. (2011). It is also predicted by equation (2b), because of the strong temperature dependencies of both K (increasing) and η* (decreasing).

Wang et al. (2016, 2017) noted that the optimal value of χ should also depend on elevation, acting through the effects of changing atmospheric pressure on the partial pressures of both oxygen, which competes with CO_2 for the Rubisco catalytic sites, and water vapour. Wang et al. (2017) used a large data set of leaf stable carbon isotope ($\delta^{13}C$) measurements to show that all three environmental dependencies are correctly predicted by the model. The predicted partial derivatives of $\ln \chi/(1-\chi)$ are 0.055 K^{-1} for temperature, –0.5 for ln vpd, and –0.08 km^{-1} for elevation. These partial derivatives were independently estimated from the $\delta^{13}C$ data by multiple linear regression, yielding 95% confidence intervals that include the predicted values: (0.046, 0.058) for temperature, (–0.61, –0.48) for ln vpd, and (–0.13, –0.08) for elevation.

4.1.2. PREDICTING GPP WITH THE CO-ORDINATION HYPOTHESIS

Wang et al. (2017) also applied the co-ordination hypothesis, which proposes that acclimation (on time scales of weeks to months) should tend to equality of Rubisco- and light-limited photosynthetic rates. This long-standing idea is well supported by independent studies (Haxeltine and Prentice, 1996; Dewar, 1996; Maire et al., 2012) and has a number of implications that are useful for modelling. These include a simple method to predict the spatial and temporal acclimation of V_{cmax} as a function of IPAR and temperature, meaning that V_{cmax} does not have to be specified independently. A variant of this principle is already included in the widely used Lund-Potsdam-Jena (LPJ) dynamic global vegetation model (Sitch et al., 2003) and models derived from LPJ, including the LPX global carbon cycle model (Stocker et al., 2012), although its implications have not been much explored by the users of these models.

Wang et al. (2017) further showed that a cost-benefit analysis of the maximum electron transport capacity J_{max} – which can be measured in the field by artificially increasing c_a to a high level – leads to a predictable optimal ratio of J_{max} to V_{cmax} that declines steeply with growth temperature, in accordance with experimental findings. The mathematical optimization was performed using the Smith formula relating the electron transport rate to absorbed PAR at the leaf level. Inclusion of this acclimation of J_{max} has been found to exert a modest but significantly beneficial effect on the prediction of V_{cmax}. Similar results can be demonstrated using the alternative empirical light response curve (a non-rectangular hyperbola with curvature parameter Θ) that is more commonly used in conjunction with the FvCB model. This alternative has not yet been implemented in the P model. Regardless of which light-response curve is used, the practical consequence is the J_{max}, like V_{cmax}, does not need to be independently specified.

Together, the elements described above define a model to predict GPP. This is achieved simply by equating the light- and Rubisco-limited rates of photosynthesis in the FvCB model (implicitly over an acclimation period of days to weeks, compatible with the time scale of remotely sensed fAPAR products), and re-arranging to eliminate χ, V_{cmax} and J_{max} (Wang et al., 2017):

$$GPP = \phi_0(C_3) \times fAPAR \times IPAR \times m \sqrt{\{1 - (c^*/m)^{2/3}\}} \quad (3a)$$

where $\phi_0(C_3)$ is the dimensionless intrinsic quantum efficiency of C_3 photosynthesis (taken to be 0.085 by Wang et al., 2017), c^* is a parameter representing the unit cost of maintaining the capacity for electron transport, and

$$m = \{c_a - \Gamma^*\} / \{c_a + 2\Gamma^* + 3\Gamma^* \sqrt{[1.6\, \eta^*\, D\, \beta^{-1}\, (K + \Gamma^*)^{-1}]}\} \quad (3b)$$

Equation (3) has the mathematical form of a LUE model: that is, for a given set of environmental conditions (ambient atmospheric CO_2, temperature, atmospheric pressure and vpd) modelled GPP is proportional to the absorbed PPFD. But unlike other LUE models, equation (3) is now explicitly defined in terms of the FvCB model of photosynthesis. Although GPP at time scales of minutes to hours (as seen, for example, during the diurnal cycle of CO_2 flux) has a well-known *saturating* response to IPAR, GPP at longer (e.g. weekly) time scales has a *linear* response to IPAR, conferred by the acclimation of V_{cmax}. This principle was previously articulated by Haxeltine and Prentice (1996) and Dewar (1996), and provides a theoretical underpinning for LUE models (Medlyn, 1998). Equation (3) gives mathematical expression to the principle, and has proved to be at least as effective in terms of simulating flux-derived monthly GPP as other LUE models – as shown in Wang et al. (2017), and in Table 2 below.

Table 2: Goodness of fit (R^2) and root-mean-squared error of prediction (RMSE) statistics for P model (Wang et al., 2017) predictions of monthly GPP, compared with results from several LUE models tested against flux measurements by Yuan et al. (2014).

	R^2		RMSE	
	This study	Yuan et al.	This study	Yuan et al.
All ecosystems	0.551	0.553 ± 0.096	2.094	2.428 ± 0.275
Shrubland	0.772	0.255 ± 0.175	2.165	1.866 ± 0.915
Deciduous broadleaf forest	0.588	0.703 ± 0.094	2.766	2.919 ± 0.450
Evergreen broadleaf forest	0.341	0.119 ± 0.063	2.046	2.961 ± 0.801
Evergreen needleleaf forest	0.535	0.501 ± 0.108	1.856	2.384 ± 0.437
Grassland	0.572	0.631 ± 0.076	2.025	2.109 ± 0.280
Mixed forest	0.700	0.637 ± 0.068	1.824	2.339 ± 0.325

4.1.3. EFFECTS OF CO_2 IN THE P MODEL

It follows from the co-ordination hypothesis that the benefit of rising CO_2 in increasing the LUE of GPP by C_3 plants will be limited to its effect on the *light-limited* rate of photosynthesis. This effect is predicted by the P model, with no additional parameter requirements, including its well-known interaction with temperature and vpd. Effects of CO_2 on different photosynthesis metrics, as

measured in 12 FACE experiments, were the subject of a meta-analysis by Ainsworth and Long (2005). They showed (for an increase of approximately 200 ppm in c_a) that LUE changed by an average of +12 ± 9%, instantaneous water-use efficiency by +54 ± 17%, and stomatal conductance by –20 ± 3%. Corresponding predictions with the P model were +17%, +55% and –15% (Wang et al., 2017).

4.1.4. SOIL MOISTURE EFFECTS

In common with many LUE models, including those used operationally, the P model does not take account of soil moisture effects – except in so far as they are manifested through changes in fAPAR. (With LST used as a driver, the P model also takes account of the effect of restricted transpiration on LST.) Analysis of flux-based GPP measurements at most of the sites considered by Wang et al. (2017), including a number of sites with pronounced dry seasons, has shown that there is no general fall-off of LUE with drought. The implication is that drought-induced reduction of GPP is already accounted for through the response of fAPAR to drought. However, this is not universally true. Some ecosystems (for example, some tropical savannas and Mediterranean forests) show reduced LUE during part or all of the dry season (Stocker et al., 2018). Moreover, extreme droughts to which ecosystems are not well adapted are expected to suppress LUE by a combination of reduced χ and reduced V_{cmax}, as has been widely observed in drying-down experiments (Zhou et al., 2013). Therefore, in common with other remotely sensed products, the current P model implementation is likely to overestimate dry-season GPP in some ecosystems and to underestimate the effects of extreme droughts on GPP. This deficiency could in principle be corrected through the use of a soil water index, in combination with the empirical functions presented by Stocker et al. (2018), to modify the modelled GPP.

4.1.5. C_4 PHOTOSYNTHESIS

The simplest way to implement C_4 photosynthesis makes just two modifications to equation (3). First, a generic ϕ_0 value suitable for C_4 plants must be chosen (taken to be 0.055 in current work). Second, c_a is made arbitrarily large. These two changes lead to a simplified equation for C_4 photosynthesis:

$$\mathrm{GPP} = \phi_0(\mathrm{C_4}) \times \mathrm{fAPAR} \times \mathrm{IPAR} \qquad (4)$$

GPP of C_4 plants can benefit from rising CO_2 under conditions of limited water availability, because water-use efficiency increases even if photosynthesis does not. However, this benefit is expected to be fully realized in increasing fAPAR.

4.1.6. MODELLING ABOVE-GROUND BIOMASS PRODUCTION

The translation from GPP to ABP (in carbon units, easily modified to dry matter units) can be summarized by the formula ABP = $(1 - f_{BG})$ x CUE x GPP, where f_{BG} is the fraction of NPP allocated below ground (including root exudation, as well as allocation to the maintenance and turnover of roots) and CUE is the carbon use efficiency, i.e. the ratio of NPP to GPP. (This formula disregards the fraction of NPP allocated to VOC emission, which is much smaller than f_{BG}.)

The additional terms required to calculate ABP from GPP are much less well understood from a theoretical and quantitative point of view than the terms in the equations for GPP itself. However, there is evidence for two competing effects of temperature on the ratio of ABP to GPP. On the one

hand, acclimation of V_{cmax} results in a weakly increasing V_{cmax} with growth temperature (Atkin et al., 2015); and leaf dark respiration varies approximately in proportion to V_{cmax}, according to the FvCB model. This effect, by itself, would cause CUE to decrease with increasing temperature. On the other hand, warm conditions increase the availability of soil nutrients by enhancing the rates of microbial metabolism, thus diminishing the need for plants to allocate carbon below ground – and thereby reducing f_{BG} (Gill and Finzi, 2017).

A preliminary statistical analysis of the NPP data provided by Michaletz et al. (2014) yielded the equations NPP ≈ 0.542 GPP + 0.0127 (T_C – 25) and ABP ≈ 0.535 GPP + 0.0074 (T_C – 25), where T_C is the growth temperature in ˚C (unpublished analysis by H. Wang). These preliminary results suggest that the effect of increased nutrient availability with temperature dominates over the effect of increased leaf respiration rate. There is substantial scatter around these relationships that likely reflects variations in soil fertility and management regimes (Vicca et al., 2012; Campioli et al., 2015). However, this general approach – applied to an improved data set under development by the University of Antwerp group – will allow us to derive an empirically well-founded, albeit approximate, prediction scheme for ABP. It is expected that soil fertility, indexed by soil pH and/or soil C:N ratio, will be a factor in the scheme.

The general approach outlined above is preferred to trying to explicitly model total autotrophic respiration, which has proved to be a major limitation of the MODIS NPP product. Our approach implicitly assumes that because of the ubiquitous acclimation of autotrophic respiration to temperature, the instantaneous response of respiration rates to temperature (as expressed in the Q_{10} factor employed by many models, including MODIS NPP) is largely irrelevant to predicting the ratio of either ABP or NPP to GPP – just as the instantaneous responses of photosynthetic rates to light and temperature do not determine the responses of actual photosynthetic rates under field conditions.

4.2. DATA NEEDS TO IMPLEMENT THE P MODEL

Here we list the data needed for implementation of the P model, and the specific data sources that were used for calibration and validation.

- **Solar radiation and vapour pressure:** For calibration, in-situ measurements at the flux sites were used. For validation, the data were obtained from the ECMWF high-resolution forecast model. These data are operational forecasts for the next 24 hours, from the ERA-Interim reanalysis (Dee et al., 2011) for 1989-2008 and thereafter obtained via the MeteoGroup operational forecast system: see https://www.ecmwf.int/sites/default/files/elibrary/2015/16559-user-guide-ecmwf-forecast-products.pdf
  IPAR data were derived from global radiation and converted to PPFD for input to the P model using the conversion factor 2.04 µmol PAR J^{-1} (Meek et al. 1984). Vapour pressure deficit was calculated from vapour pressure and temperature by standard equations in Allen et al. (1998).
- **Temperature:** For calibration, daily temperature data were provided in ATBD v1 from in-situ air temperature measurements at the flux sites, and here from remotely sensed daytime LST. For validation, daily temperature data were provided from (a) ECMWF meteorological data, as described above for solar radiation and vapour pressure, and (b) again from remotely sensed daytime LST. The source of remotely sensed LST data was level 2 ENVISAT AATSR (Ghent et al., 2012) obtained via the ESA project Globtemperature (http://www.globtemperature.info/). Pre-processing of these data involved cloud/shadow masking and interpolation and smoothing (modified from Swets et al., 1999) to provide 10-daily averages.

- **fAPAR** data for calibration were obtained from SeaWiFS and MERIS GVI data (Ceccherini et al., 2013) via fapar.jrc.ec.europa.eu. For validation, fAPAR data were obtained from MERIS GVI (Gobron et al., 1999) via http://meriss10.vgt.vito.be (fAPAR data will subsequently be obtained from Sentinel-3.) Values of fAPAR were multiplied by 0.8 to account for incomplete utilization of the full spectrum of photosynthetically active radiation (400-700 nm) by photosynthesis (von Caemmerer, 2000).
- **Ambient partial pressure of CO_2**: the time-varying CO_2 mole fraction obtained from the monitoring station at Mauna Loa, Hawaii is converted to partial pressure units, and used as input to the P model. Data are from the Scripps Institution CO_2 monitoring network (Keeling et al., 2001):
  http://scrippsco2.ucsd.edu/data/atmospheric_co2/mlo
- **C_3 versus C_4 photosynthesis**: the need for data on the distribution of C_3 versus C_4 plants (which is problematic, especially where crops are concerned) will be circumvented, by providing both values for every pixel.

Subsequent developments considered for later implementation are:

- Inclusion of a **soil-moisture** effect on LUE. In relatively moist soils the actual soil moisture content has little or no effect on LUE. However, very low soil moisture can negatively affect LUE in addition to the effect of high vpd (as noted in section 4.1.4 above, and by Stocker et al., 2018). This effect is not incorporated in existing operational products which, therefore, share a common bias towards overpredicting GPP under drought conditions. This bias could be removed by using a remotely sensed soil water index as an additional driver of the P model.
- Use of either a remotely sensed **digital elevation model**, or remotely sensed surface pressure, to account for the various atmospheric pressure effects on photosynthesis via the P model. This would improve the accuracy of modelled GPP values at high elevations.
- For the ABP model: use of **global soils data**, which could be translated into scalars reflecting the effect of soil fertility on the ratio of ABP to GPP. Soil fertility can be indexed e.g. by pH (low-pH soils tend to be less fertile) or C:N ratio (more fertile soils tend to have narrower C:N ratios). Machine learning methods have been used to combine national soil survey mapping with soil profile measurements at 250 m resolution (https://www.soilgrids.org) and provide the best available, albiet imperfect, source of global data on soil types and properties.

4.3. THE APPROACH TO ESTIMATING PER-PIXEL UNCERTAINTY IN GPP

Two independent methods will be applied to generate per-pixel uncertainties, taking into account that on the one hand, the P model algorithm is derived from first principles and consists of a single equation, which can be differentiated with respect to all of the uncertain quantities that it contains; and on the other hand, the algorithm's credibility is assured by its ability to predict independently measured GPP, which will be quantified.

Thus two methods will be used to quantify uncertainties in GPP. The first method, which we have implemented while generating outputs for validation, is a classical Type B uncertainty evaluation: derived analytically and producing a per-pixel uncertainty value explicitly considering the known sources of uncertainty in different quantities entering the model and combining them using established principles. The second method can be considered as a Type A uncertainty evaluation in so far as data from different observation periods at a flux measurement site, and data from flux different sites, can be considered as stochastic realizations of the same underlying processes. Uncertainty estimates obtained by the second (Type A) method may include consequences of processes that are not explicitly included in the GPP algorithm, potentially leading to wider

uncertainty estimates than those obtained by the first (Type B) method. A criterion for this approach to work well is that the estimated uncertainty should be based on a sufficiently large and diverse ensemble of flux sites. Implicitly, errors in flux measurements and their partitioning to GPP would be included in the Type A uncertainty assessment.

4.3.1. UNCERTAINTY EVALUATION BASED ON THE P MODEL ALGORITHM

Equations (3) and (4) contain a number of input variables and parameters whose uncertainty can be quantified. In addition, a number of the photosynthetic parameters are temperature-dependent. Uncertainties in the temperature dependencies are separated from uncertainty in the temperature data by applying the following standard formulae:

$$\Gamma^* = \Gamma^*[25] \exp\{(\Delta H_{\Gamma^*}/R)(1/298.15 - 1/T)\} \quad (5)$$

$$\eta^* = \exp\{580\,[1/(T - 138)] - [1/(160]\} \quad (6)$$

$$K = K_C\,(1 + O/K_O) \quad (7)$$

$$K_C = K_C[25] \exp\{(\Delta H_{K\_C}/R)(1/298.15 - 1/T)\} \quad (8)$$

$$K_O = K_O[25] \exp\{(\Delta H_{K\_O}/R)(1/298.15 - 1/T)\} \quad (9)$$

where R is the universal gas constant (8.314 46 J mol^{-1} K^{-1}), T is the canopy temperature (K), K_C is the Michaelis-Menten coefficient for carboxylation (Pa), K_O is the Michaelis-Menten coefficient for oxygenation (Pa), O is the partial pressure of oxygen (209 460 µmol mol^{-1} x atmospheric pressure in Pa); $\Gamma^*[25]$, $K_C[25]$ and $K_O[25]$ are the values of Γ^*, K_C and K_O, respectively, at 298.15 K; and ΔH_{Γ^*}, ΔH_{K_C} and ΔH_{K_O} are the corresponding activation energies (J mol^{-1}). Moreover, if D is estimated from absolute water vapour pressure (e_a) and saturation vapour pressure (e_s), then:

$$D = e_s(T_C) - e_a \quad (10)$$

where $e_s = e_s(0) \exp\{17.27\,T_C/(T_C + 237.3)\}$ (Pa) and $T_C = T - 273.15$ K.

Those quantities that are either defined precisely, or known with an uncertainty that is effectively negligible in this context, have been assigned numerical values above and will not be considered further. In the following section, we describe the approach that we have adopted in the validation exercise to derive standard uncertainties for the remaining quantities. The formulation above allows each of the sources of uncertainty to be considered independent and, therefore, uncertainties from each source to be combined using the standard formula:

$$u^2(y) = \sum_i (\partial f/\partial x_i)^2\, u^2(x_i) \quad (11)$$

where $u(y)$ is the standard uncertainty of GPP, $\partial f/\partial x_i$ is the sensitivity of GPP to variable x_i (obtained by differentiating equation (1) with respect to each uncertain variable and evaluating the partial derivative at the current central value of x_i), and $u(x_i)$ is the standard uncertainty of x_i.

4.3.2. DATA UNCERTAINTIES

fAPAR: standard uncertainties have been estimated as the standard deviation of values in the 9 x 9 pixel grid surrounding the pixel of interest.

LST: the data have been smoothed and gap-filled. Uncertainties provided on a per-pixel basis have been averaged over each dekad for each pixel.

CO_2 mole fraction data, obtained on an annual basis from the Scripps Institute of Oceanography Mauna Loa record, have been assigned a nominal uncertainty of ± 0.1 ppm. However, substantially greater uncertainty derives from local variations in c_a due to ground-level sources and sinks in soils and vegetation, and local industrial and/or transport sources. This uncertainty was approximated as the difference between the current global value and the corresponding measured value at each of the flux sites. The total uncertainty ascribed to CO_2 was small, < 0.1%.

Uncertainties were not available for data on **shortwave radiation** or **vapour pressure**, and are not provided in the ECMWF operational data stream. However, we note that the calculation of vpd depends on LST and therefore the uncertainties in LST, at least, propagate into vpd.

In the global GPP product, uncertainties on fAPAR will be derived from per-pixel uncertainties provided with the Sentinel-3 data. How to estimate uncertainties in the other meteorological variables remains to be considered.

Parameter uncertainties

The parameter **β** was estimated based on leaf $\delta^{13}C$ data, from the intercept of the regression of ln $\chi/(1-\chi)$ against environmental predictors (Wang et al., 2017). The uncertainty of this estimate was assessed from the standard error of the intercept, and inflated to account for uncertainty in the conversion from stable isotope measurements to χ. The value used was β = 146 ± 2.7 (H. Wang, unpublished analysis).

The parameter ***c**** was estimated from published values of electron transport capacity (J_{max}) and carboxylation capacity (V_{cmax}) under a variety of experimental growth conditions (Kattge and Knorr, 2007; Wang et al., 2017). The uncertainty of *c** was estimated based on a regression of experimentally determined J_{max}/V_{cmax} values against growth temperature. The value used was 0.41 ± 0.112 (H. Wang, unpublished analysis).

The remaining parameters of equations (3) and (4) are standard elements of the FcVB photosynthesis model. They are rather accurately measured, and show relatively little variation among different plant species and measurement techniques. Nonetheless, they are subject to some uncertainty, which is taken into account as described below.

$\phi_0(C_3)$ and $\phi_0(C_4)$: published surveys of measurements on various species show some variability in these parameters (Skillman, 2008; Zhu et al., 2010), with an approximately normally distribution across species within each photosynthetic pathway. It is appopriate to calibrate these parameters within a plausible range, because of natural variation in their values across species; natural variation in the fraction of photosynthetically active rdaiation used for photosynthesis; and unresolved systematic variation in magnitude among different remotely sensed fAPAR products.

Γ*[25], K_C[25], K_O[25] and the corresponding activation energies: most recent modelling studies have used the *in vivo* values determined originally by Bernacchi et al. (2001), but other experimental data sets have given slightly different reference values and activation energies (De Kauwe et al., 2016b). There is also some variation in Rubisco kinetic properties across species from different environments (e.g. Hermida-Carerra et al., 2016). Based on this literature, we have adopted the following values (and uncertainties) for each parameter: Γ*[25] = 4.08 ± 0.10 Pa at standard atmospheric pressure; $\Delta H_{\Gamma*}$ = 27055.67 ± 5020.93 J mol^{-1}; K_C[25] = 40.41 ± 3.45 Pa; ΔH_{K_C} =

64805.5 ± 5018.50 J mol^{-1}; $K_O[25]$ = 27480 Pa (no uncertainty assigned); ΔH_{K_O} = 36164 ± 152.74 J mol^{-1}.

4.3.3. COMBINING UNCERTAINTIES

Derivatives of equation (1) with respect to each uncertain quantity have been obtained analytically. For constant quantities such as the two ϕ_0 values the derivative can be pre-calculated. For quantities that vary in time and/or space, the derivative will be evaluated as part of the standard workflow. The outputs of the workflow per time-step and pixel will include the central estimate of GPP from equation (1) and its composite standard uncertainty from equation (2) with elements calculated in the manner described above.

4.3.4. UNCERTAINTY EVALUATION BASED ON A COMPARISON OF MODELLED AND MEASURED GPP

Sections 4.3.1 to 4.3.3 above outline the method adopted for the calculation of a Type B, per-pixel uncertainty for modelled GPP. The alternative Type A approach involves defining a model for total uncertainty based on a statistical comparison of observed and modelled GPP across space and time. The comparative analysis of modelled and measured GPP by Wang et al. (2017) indicates a lack of bias, and visually suggests that a suitable model for the empirical distribution of uncertainties could consider relative uncertainty (fraction of estimated GPP) as a constant. This conclusion will be re-visited based on the new evaluation in TerrA-P, and used to define a Type A uncertainty calculation. The two proposed methods to quantify uncertainty will be complementary and act as a cross-check on one another.

4.4. A PRELIMINARY CALIBRATION DATA SET FOR GPP

The University of Antwerp group has selected from the most recent synthesis data set (http://fluxnet.fluxdata.org/data/fluxnet2015-dataset/) 17 flux sites in different biomes that have data in the public domain, and are characterized by multi-year records (daily data over a period of at least 5 years) with good quality GPP data, checked by the standardized methodology defined by FLUXNET (Reichstein et al., 2005; Papale et al., 2006), and a large, relatively homogeneous vegetation footprint (at least 1 km x 1 km) to ensure reliable comparisons between *in situ* and remotely sensed data. These 17 sites (Table 3) provide the basis for calibration.

Table 3: The calibration set of eddy-covariance flux measurement sites. VEG = IGBP vegetation type: EBF = evergreen broadleaf forest, ENF = evergreen needleleaf forest, OSH = open shrubland, CRO = cropland, DBF = deciduous broadleaf forest.

CODE	NAME	LAT (˚)	LONG (˚)	ELEV (m)	VEG
AU-Tum	Tumbarumba	–35.6566	148.1517	645	EBF
CA-NS3	UCI-1964 burn site	55.9117	–98.3822	260	ENF
CA-NS6	UCI-1989 burn site	55.9167	–98.9644	244	OSH
CA-Obs	Saskatchewan – Western Boreal, Mature Black Spruce	53.9872	–105.1178	629	ENF
DE-Geb	Gebesee	51.1001	10.9143	162	CRO
DE-Hai	Hainich	51.0792	10.4530	430	DBF
DE-Kli	Klingenberg	50.8929	13.5225	478	CRO
FI-Hyy	Hyyttiälä	61.8475	24.2950	181	ENF
FR-Fon	Fontainebleau-Barbeau	48.4764	2.7801	103	DBF
FR-LBr	Le Bray (after 28 June 1998)	44.7171	–0.7693	61	ENF
FR-Pue	Puechabon	43.7414	3.5958	270	EBF
IT-Cpz	Castelporziano	41.7052	12.3761	68	EBF
NL-Loo	Loobos	52.1666	5.7436	25	ENF
US-Ha1	Harvard Forest EMS Tower (HFR1)	42.5378	–72.1715	340	DBF
US-MMS	Morgan Monroe State Forest	39.3232	–86.4131	275	DBF
US-UMB	University of Michigan Biological Station	45.5598	–84.7138	234	DBF
US-WCr	Willow Creek	45.8059	–90.0799	520	DBF

4.5. CALIBRATION RESULTS

Simulations were set up for each of the calibration sites using local meteorological measurements of daily total incoming shortwave radiation and vapour pressure. Temperature was derived from remotely sensed LST, and vapour pressure was converted to vpd using the standard method (Allen et al. 1998) using LST as the relevant temperature for saturation vapour pressure. Annual values of CO_2 were prescribed. Low-temperature inhibition of photosynthesis was represented in the simplest possible way, by setting GPP to zero during periods with subfreezing temperatures.

$\phi_0(C_3)$ was estimated from the comparison of P model estimates with the GPP data by varying its value in the model between 0.05 and 0.1125. The optimized value (yielding the smallest sum of RMSE across sites) obtained in the initial calibration (ATBD v1) was 0.084: only marginally different from the value of 0.085 adopted in Wang et al. (2017). The updated calibration presented here, with temperature provided by LST, yielded an optimized value of 0.092, which is still well within the experimentally observed range based on leaf-level measurements (Skillman, 2008). Figure 2 shows the effect of varying $\phi_0(C_3)$ on the sum of RMSE across sites in the updated calibration.

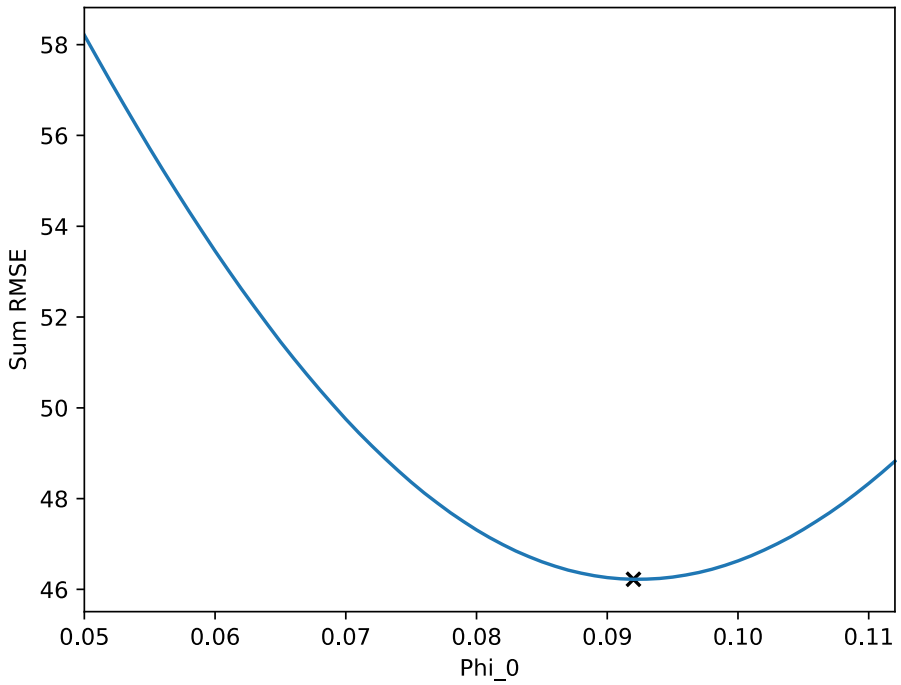
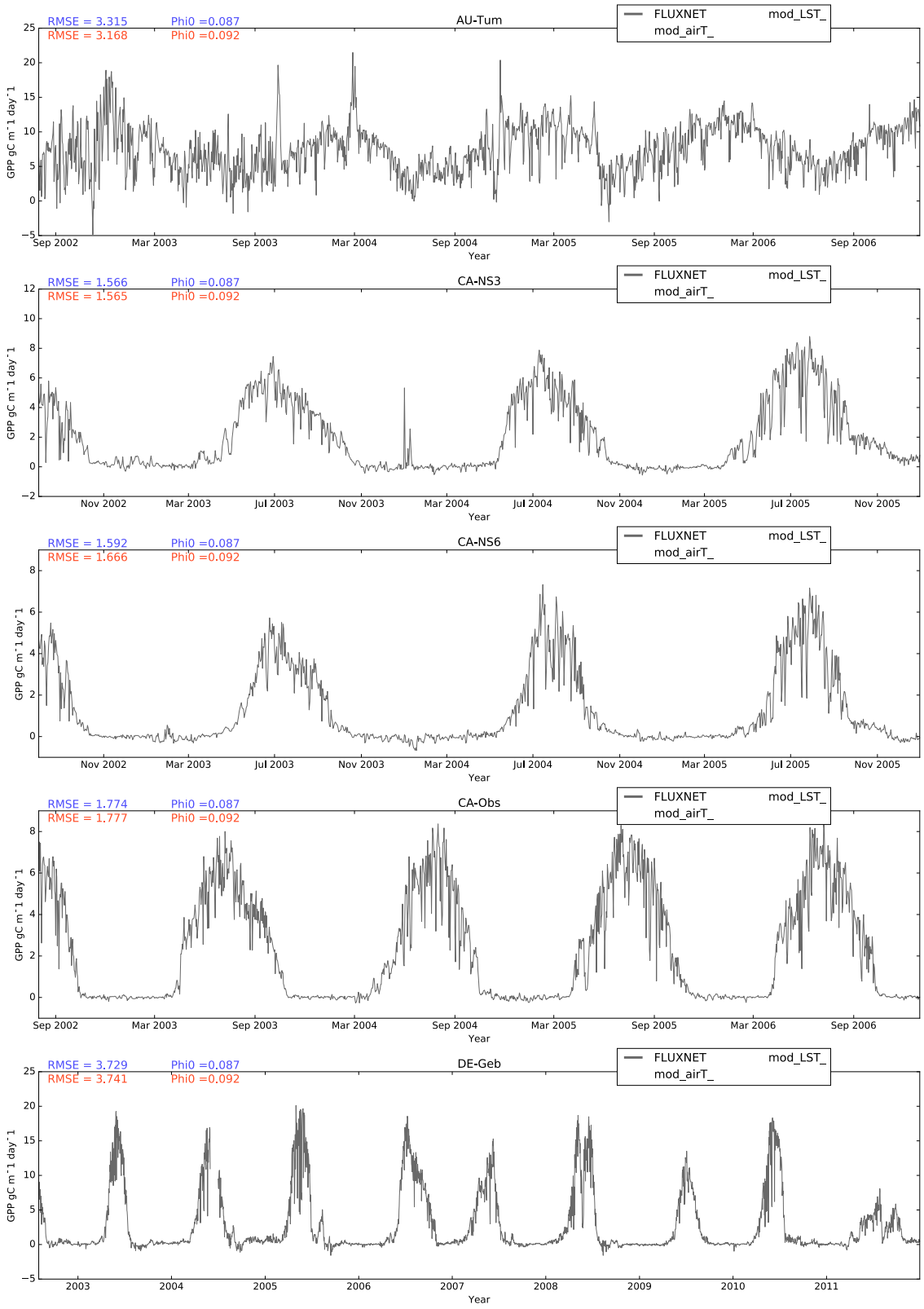
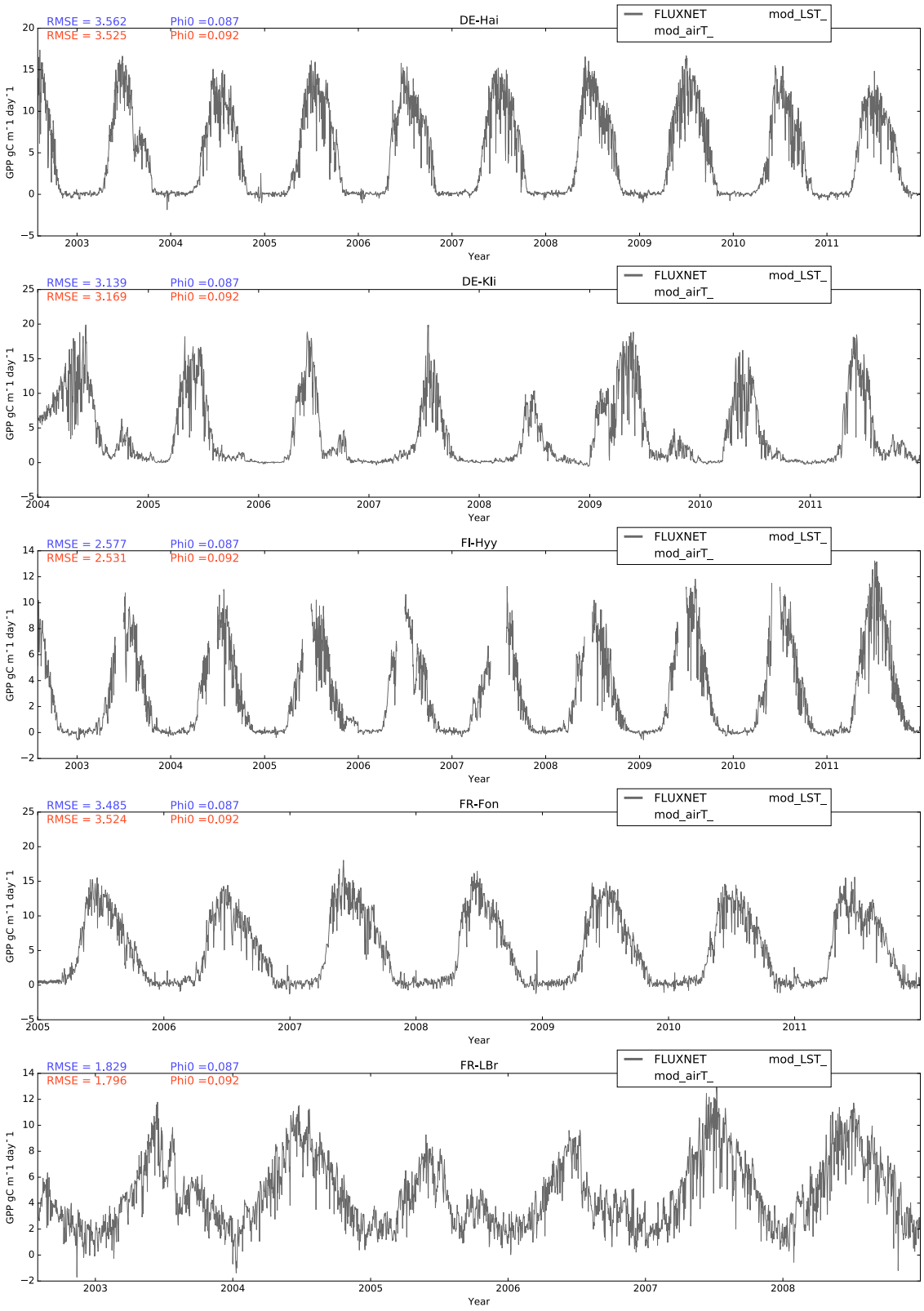


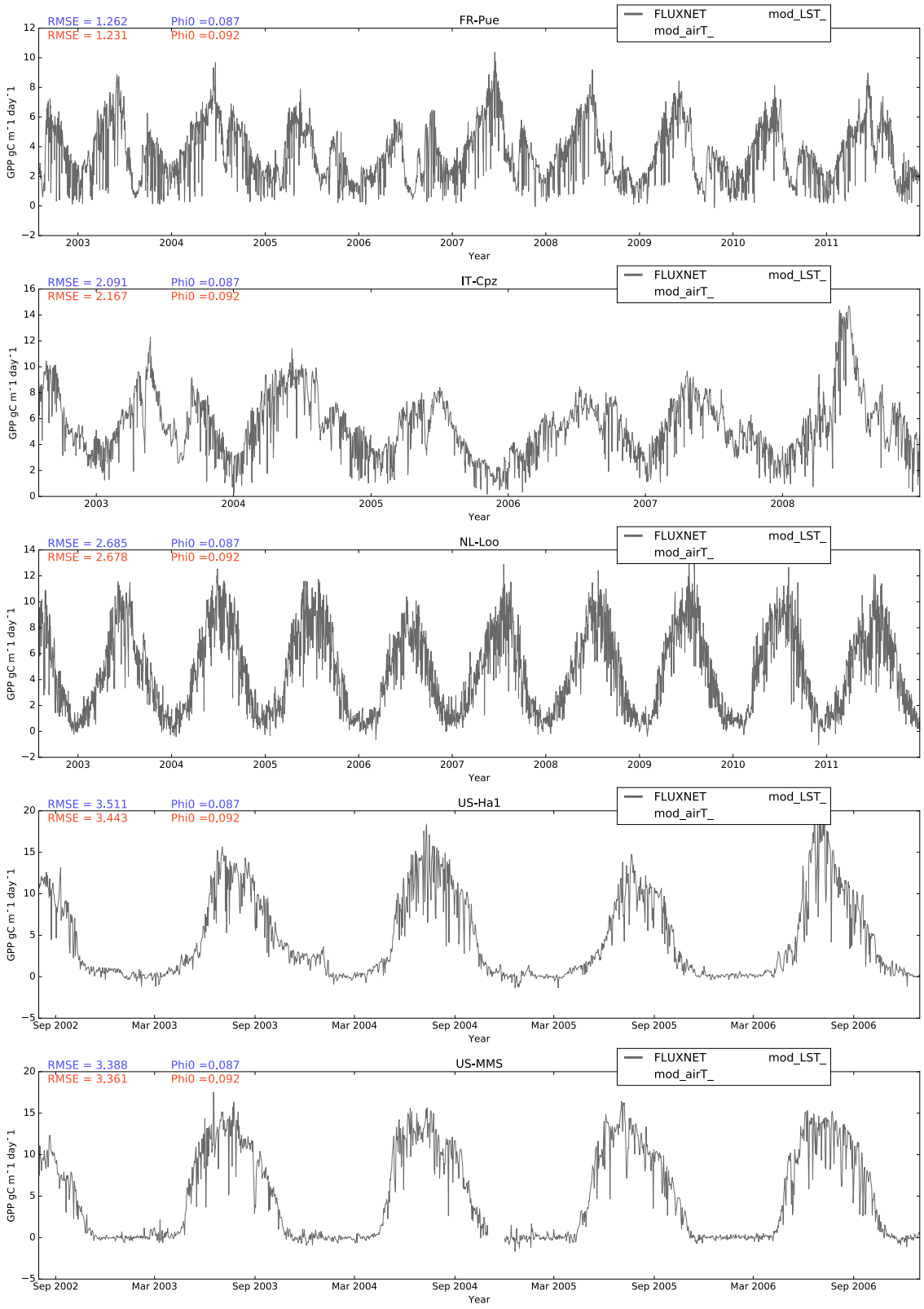
Figure 2: Effect of varying $\varphi_0(C_3)$ on the summed daily RMSE between flux-derived and modelled GPP at the 17 calibration sites.

$\phi_0(C_4)$ was not calibrated but should be estimated using data from C_4-dominated vegetation, which was not represented in the calibration set of sites.

Figure 3 shows the results of data-model comparison in the form of time series of GPP from the flux measurements, and from the P model with optimized $\phi_0(C_3)$ according to the updated calibration. Visual agreement and RMSE values are generally satisfactory. There are some mismatches, which do not appear to be related to vegetation type. Indeed, there is no indication in this comparison that GPP is systematically either under-or overestimated in any one vegetation type. GPP is generally underestimated at AU-Tum, for unknown reasons. Mismatches include underestimation of peak-season GPP by the model at a few sites; and in some sites and years, the simulation of positive GPP around the start and/or end of the growing season at times when the flux-derived GPP is close to zero. This latter problem is alleviated, although not completely removed, by the substition of LST for air temperature as a driver. Peaks in observed GPP during winter, e.g. at CA-NS3, are not simulated but are presumed to be artefacts in the data.


AU-Tum
RMSE = 3.315 Phi0 =0.087
RMSE = 3.168 Phi0 =0.092
FLUXNET mod_LST_
mod_airT_
GPP gC m^-1 day^-1
Year
CA-NS3
RMSE = 1.566 Phi0 =0.087
RMSE = 1.565 Phi0 =0.092
CA-NS6
RMSE = 1.592 Phi0 =0.087
RMSE = 1.666 Phi0 =0.092
CA-Obs
RMSE = 1.774 Phi0 =0.087
RMSE = 1.777 Phi0 =0.092
DE-Geb
RMSE = 3.729 Phi0 =0.087
RMSE = 3.741 Phi0 =0.092


RMSE = 3.562
Phi0 =0.087
RMSE = 3.525
Phi0 =0.092
DE-Hai
FLUXNET
mod_LST_
mod_airT_
GPP gC m^-1 day^-1
Year
RMSE = 3.139
Phi0 =0.087
RMSE = 3.169
Phi0 =0.092
DE-Kli
RMSE = 2.577
Phi0 =0.087
RMSE = 2.531
Phi0 =0.092
FI-Hyy
RMSE = 3.485
Phi0 =0.087
RMSE = 3.524
Phi0 =0.092
FR-Fon
RMSE = 1.829
Phi0 =0.087
RMSE = 1.796
Phi0 =0.092
FR-LBr


RMSE = 1.262
RMSE = 1.231
Phi0 =0.087
Phi0 =0.092
FR-Pue
FLUXNET
mod_LST_
mod_airT_
GPP gC m^-1 day^-1
Year
RMSE = 2.091
RMSE = 2.167
Phi0 =0.087
Phi0 =0.092
IT-Cpz
RMSE = 2.685
RMSE = 2.678
Phi0 =0.087
Phi0 =0.092
NL-Loo
RMSE = 3.511
RMSE = 3.443
Phi0 =0.087
Phi0 =0.092
US-Ha1
Sep 2002
Mar 2003
Sep 2003
Mar 2004
Sep 2004
Mar 2005
Sep 2005
Mar 2006
Sep 2006
RMSE = 3.388
RMSE = 3.361
Phi0 =0.087
Phi0 =0.092
US-MMS

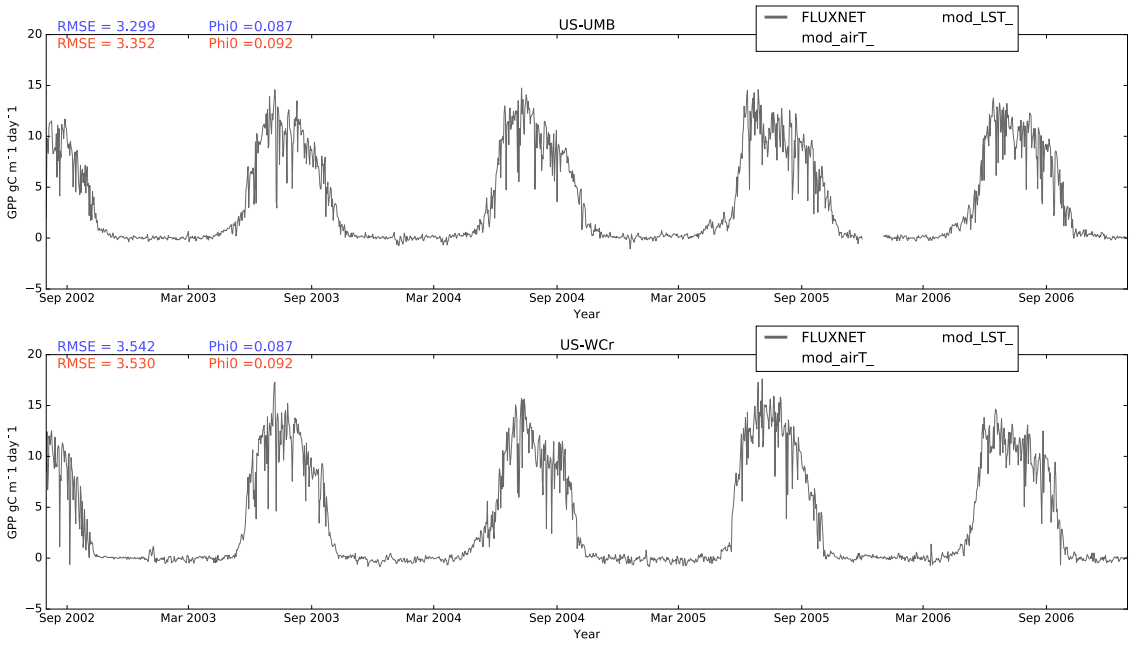
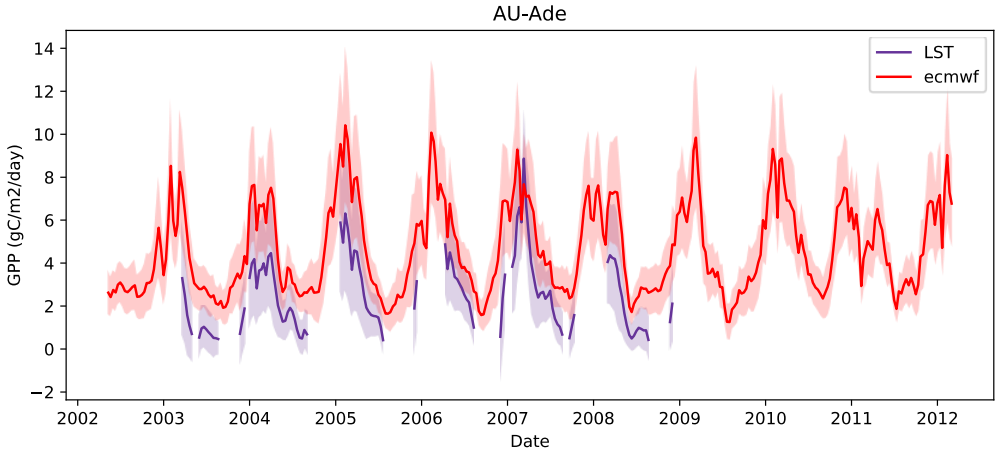
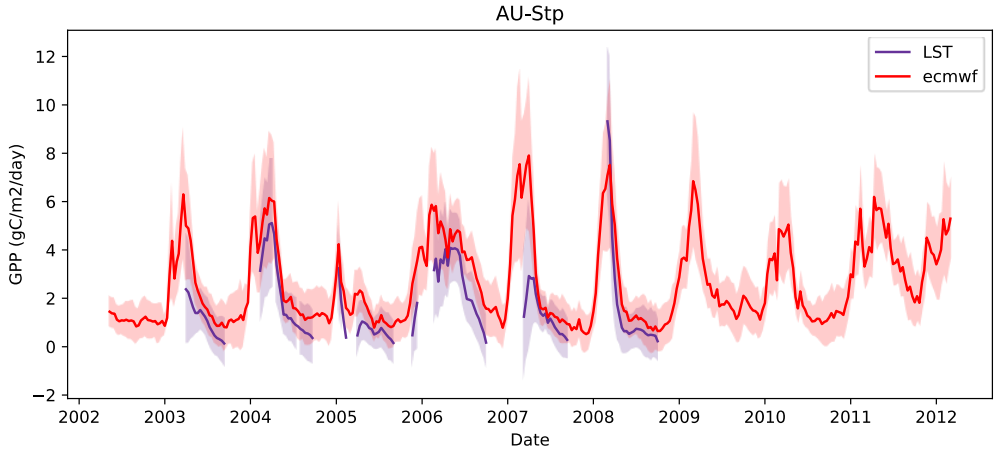
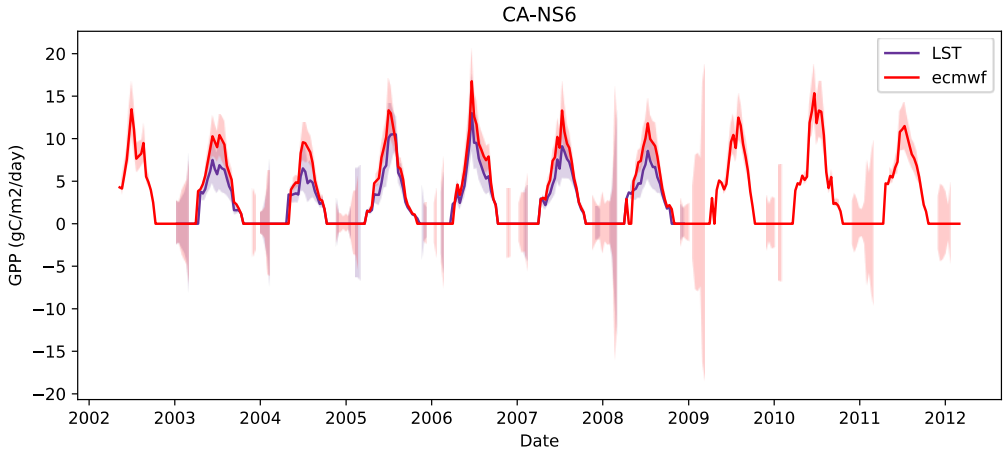
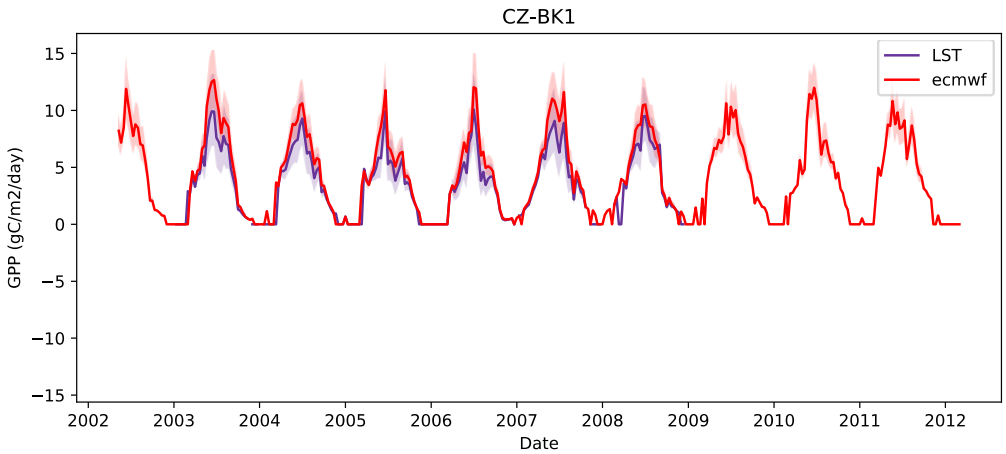


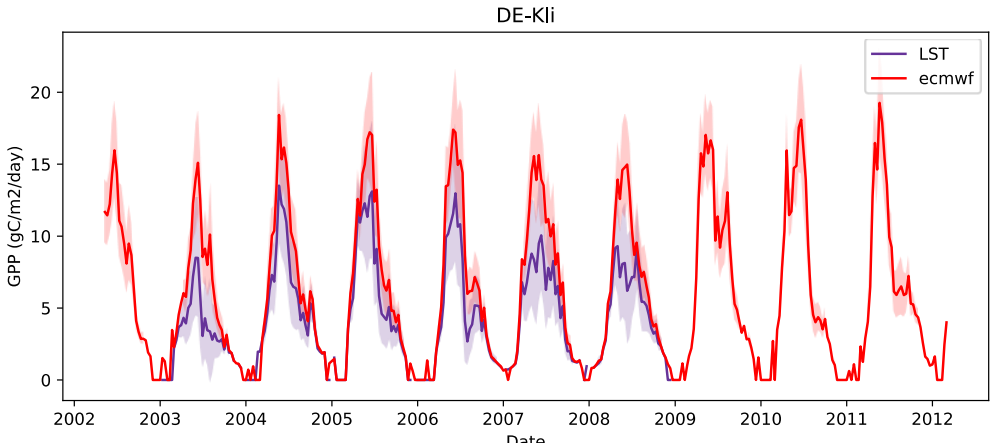
Figure 3: Comparison of flux-derived GPP and P model-simulated GPP at the calibration sites. The dark grey traces represent the mean GPP from the alternative FLUXNET partitioning methods. The red traces represent modelled GPP (updated calibration driven by LST); the blue traces represent modelled GPP driven by air temperature.

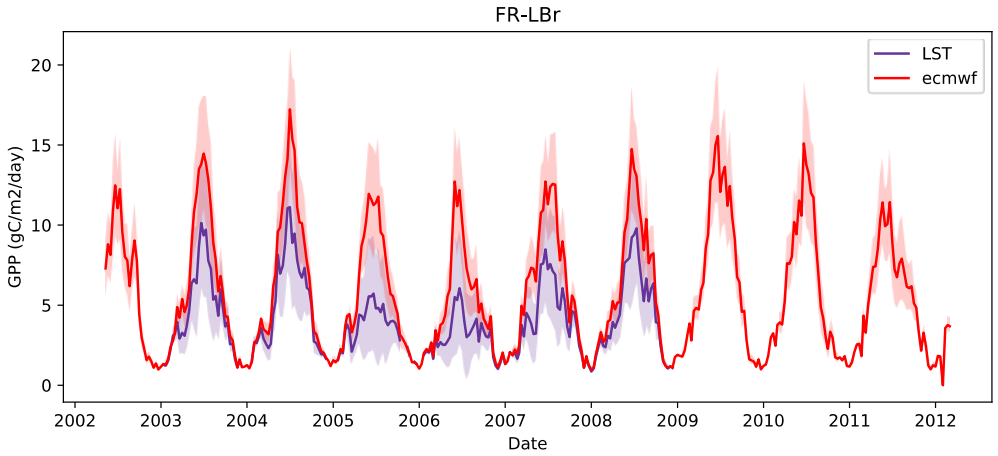


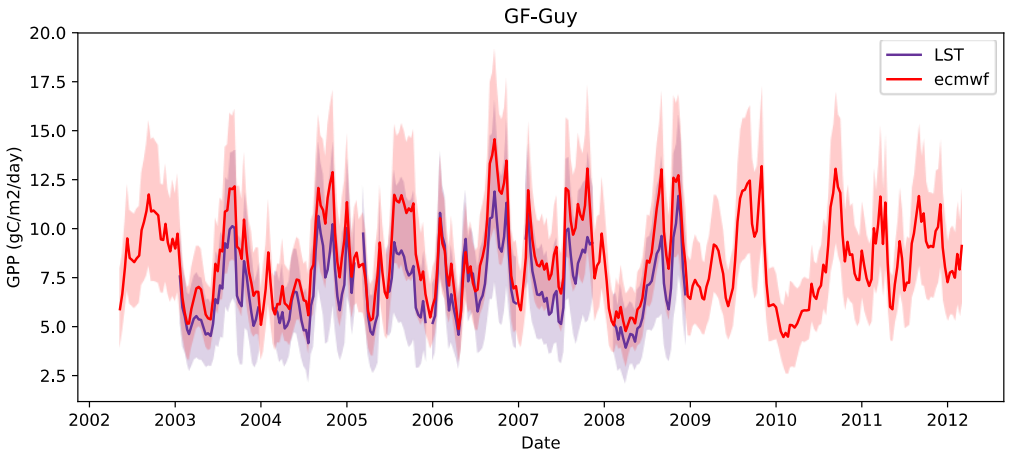

AU-Stp
LST
ecmwf
GPP (gC/m2/day)
Date


CA-NS6
LST
ecmwf
GPP (gC/m2/day)
Date


CZ-BK1
LST
ecmwf
GPP (gC/m2/day)
Date


DE-Kli
LST
ecmwf
GPP (gC/m2/day)
0
5
10
15
20
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
Date


FR-LBr
LST
ecmwf
GPP (gC/m2/day)
0
5
10
15
20
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
Date


GF-Guy
LST
ecmwf
GPP (gC/m2/day)
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
Date

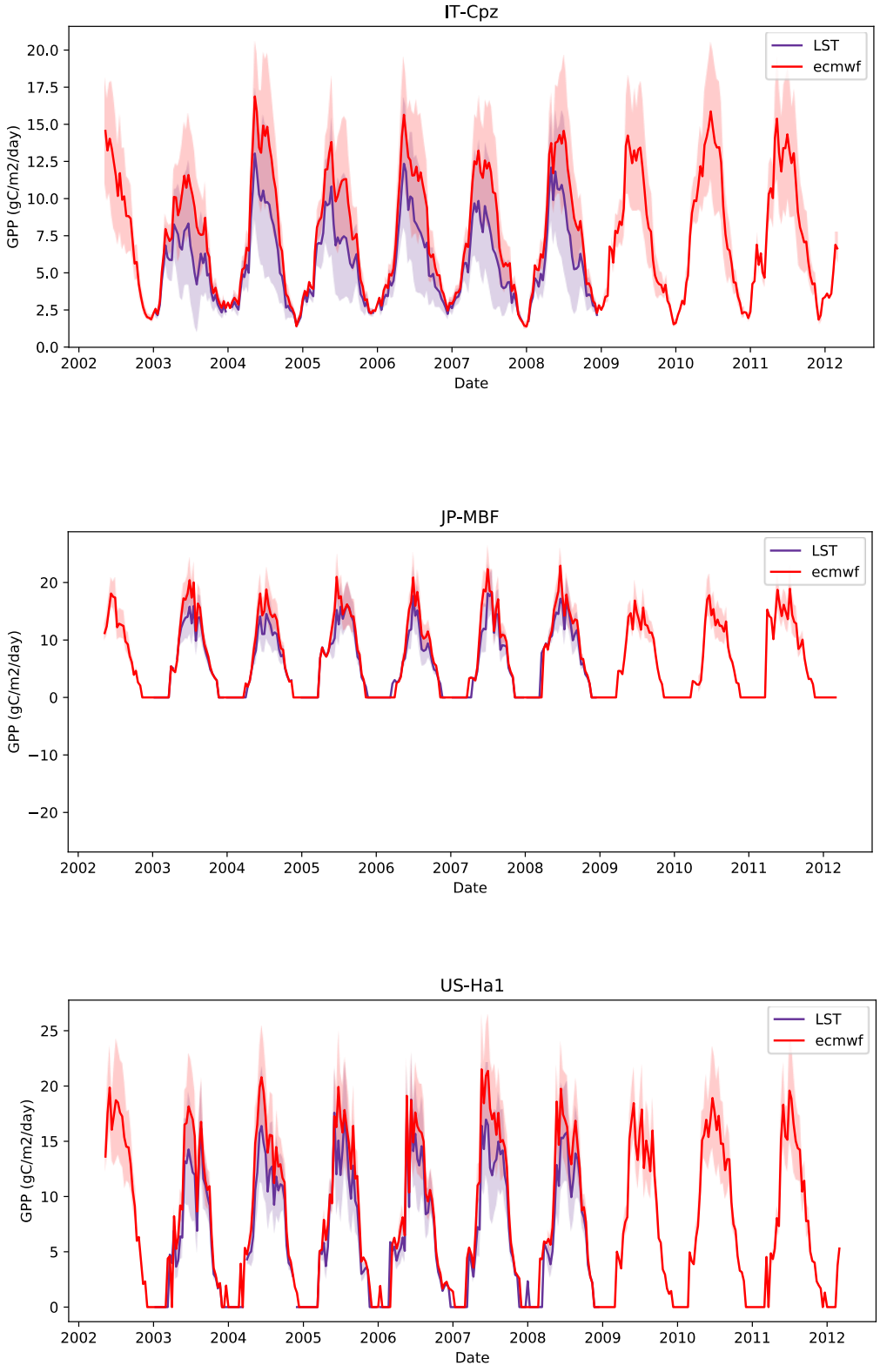


Figure 4: Examples of P model-simulated GPP outputs (with uncertainties) as provided for the validation exercise, with temperature either interpolated from ECMWF meteorological data or from remotely sensed LST.

CHAPTER 5 VALIDATION APPROACH

This chapter summarizes the validation and proposed comparative benchmarking of the products:

(1) The point location products have been validated with in-situ measurements of GPP to assess their accuracy. The same approach is proposed for ABP when available.

(2) The global spatial GPP/ABP products will subsequently be compared to other operational EO products (MODIS, C-GLOPS1) to assess how much, and where, the new products differ from the existing ones.

5.1. VALIDATION METHOD AGAINST IN-SITU DATA

Validation aims to assess the capability of model simulations to describe carbon dynamics for a variety of ecosystems and to identify potential ways to improve them. A separate validation report tests how well the GPP simulations describe annual carbon dynamics for different ecosystems and climates; encompass the daily and seasonal trends of GPP; describe interannual variability; and describe the main environmental controls of GPP.

5.1.1. DESCRIPTION OF THE IN SITU DATA

Validation of modelled GPP in the TerrA-P project relies on the FLUXNET 2015 database of values obtained with the eddy-covariance technique. This well-established technique provides GPP by post-processing of direct measurements of net ecosystem CO_2 exchange. FLUXNET includes data from regional networks, international projects and field-sites of research institutes and provides a thoroughly standardized data treatment and data analysis for sites distributed across the world. These data have been extensively used for the development and evaluation of ecological models at regional or global scale (Balzarolo et al 2014).

The FLUXNET GPP data are well suited to use as validation products for several reasons: (i) they are available at both high time resolution (half-hourly) and aggregated to longer time steps (daily to annual); (ii) they are available for multiple years (>10 years for the most intensively studied sites); (iii) they typically measure an area of the ecosystem (footprint) comparable to the resolution of remote sensed products; (iv) data are provided with uncertainty estimations; and (v) sites are available globally and for all types of terrestrial ecosystems, including forests, savannas, grasslands, croplands, wetlands and tundra. FLUXNET was established in 1998 (Baldocchi et al 2001) and since then techniques have improved and different database versions have been produced. We use the latest version (FLUXNET 2015) and confine attention to publicly available data, for which data use is free and open provided that proper acknowledgment is given to site PIs and funding agencies (Tier 1, see http://fluxnet.fluxdata.org/data/fluxnet2015-dataset/). About 200-250 GPP sites are publicly available now, of which 122 met our criteria for quality, homogeneity and length of record.

For ABP, we will use a dataset recently released by the University of Antwerp group (Vicca et al., 2012; Campioli et al., 2015). This dataset is best suited for validation purposes because (i) it provides quality-controlled data for both aboveground and belowground biomass production, (ii) it

provides standardized uncertainty estimates, (iii) it provides ecosystem level data (e.g. dominant and codominant species, overstory and understory) compatible with the spatial footprint of the FLUXNET 2015 sites, thus comparable to remotely sensed data and (iv) it provides biomass production data paralleled by GPP for the same year of measurements.

5.1.2. VALIDATION METHOD

Validation is carried out at high spatial resolution i.e. at site level; we do not plan to perform landscape or regional validations, as this would require up-scaling of in-situ data with a large propagation of uncertainty. A new fAPAR data set was applied. Otherwise, the model and its data inputs follow the same protocol as was used for the updated calibration and presented in Figures 2 and 3 above. These outputs also include uncertainties (by site and dekad) in modelled GPP calculated by the Type B method described above. Some examples of these outputs, representing major biomes, are illustrated in Figure 4.

5.2. BENCHMARK METHOD TO OTHER DATA SETS

5.2.1. REFERENCE DATA SETS

At global scale a comparison will be done with two operational EO GPP/NPP products, both discussed in Chapter 3 above:

The C-GLOPS1 Dry Matter Production (DMP) product at 10-daily time steps and expressed in kg DM ha^{-1} day^{-1}. The DMP is generated by a LUE model first implemented by Veroustraete (1994) and modified and improved in the MARSOP and Copernicus Global Land Service project to run on SPOT-VGT/PROBA-V imagery and ECMWF meteorological data. Within C-GLOPS, the product has been extensively validated against in-situ, MODIS and other modelled datasets of GPP and NPP. The ATBD and validation report of the DMP are available through the C-GLOPS website, http://land.copernicus.eu. The current online version 1 is based on 1 km SPOT-VGT and PROBA-V fAPAR data derived from the MARSOP project. In the meantime, a second version of the DMP is in development and has been validated. Besides a number of algorithmic changes, this version is based on an improved dataset of SPOT-VGT and PROBA-V fAPAR. Both products will be compared with the P-model output.

MODIS GPP (MOD17A2) and NPP (MOD17A3) as described in detail by Running et al. (1999), Heinsch et al. (2003) and Zhao et al. (2005). This is also a variant of the satellite-based LUE approach. The GPP product is available at 8-daily timesteps. The NPP product is derived on an annual basis using maintenance and growth respiration estimates linking daily biomass and annual growth of plant tissues to the satellite-derived estimates of leaf area index (LAI). The MYD17A3H Version 6 product, recently released by MODIS, provides estimated annual NPP at 500 m pixel resolution. Annual NPP is derived from the sum of the 45 8-day Net Photosynthesis (PSN) products (MYD17A2H) from the given year.

5.2.2. METHODS

The methods for the benchmarking of the different EO-derived vegetation production data sets are based on guidelines, protocols and metrics defined by the Land Product Validation (LPV) group of

the Committee on Earth Observation Satellite (CEOS) for the validation of satellite-derived land products. The following aspects will be evaluated.

(1) *Product completeness*: the missing values or pixels flagged as invalid over land were quantified, overall and over different biomes. An aggregated version of the ESA CCI land cover map will used for this purpose.
(2) Spatial consistency analysis:
   - *Spatial distribution of the GPP/NPP values*: global maps of metrics expressing the similarity and difference between different global GPP/NPP time series will be computed. The metrics include the Root Mean Squared Error (RMSE).
   - *Magnitude of the retrievals*: global yearly averages will be calculated for GPP and NPP.

(3) Global statistical analysis:
   - *Histograms of bias*: histograms of residuals between products.
   - *Distribution per biome type*: statistical distributions of GPP/NPP values and residuals, computed over biomes for the different data sets. An aggregated version of the ESA CCI land cover map will used for this purpose.
   - *Global statistics*: Scatterplots between the different datasets will be produced at a global scale and per biome. Metrics (e.g. coefficient of determination, agreement coefficient, orthogonal regression) among different data sets are computed per biome.

(4) Temporal consistency analysis
   - *Temporal variation*: Statistical metrics among different data sets are computed per scene to evaluate the time evolution of the metrics.

REFERENCES

Ainsworth, E.A. and S.P. Long (2005) What have we learned from 15 years of free-air CO_2 enrichment? A meta-analytic review of the responses of photosynthesis, canopy properties and plant production to rising CO_2. *New Phytologist* **165**: 351–372.

Allen, R.G., L.S. Pereira, D. Raes and M. Smith (1999) Crop evapotranspiration – Guidelines for computing crop water requirements. *FAO Irrigation and Drainage Paper* **56.**

Anav, A., P. Friedlingstein, C. Beer, P. Ciais, A. Harper, C. Jones, G. Murray-Tortarolo, D. Papale, N.C. Parazoo and P. Peylin (2015) Spatiotemporal patterns of terrestrial gross primary production: a review. *Reviews of Geophysics* **53**: 785–818.

Atkin, O.K., K.J. Bloomfield, P.B. Reich, M.G. Tjoelker, G.P. Asner, D. Bonal, G. Bönisch, M. Bradford, L.A. Cernusak, E.G. Cosio, D. Creek, K.Y. Crous, T. Domingues, J.S. Dukes, J.J.G. Egerton, J.R. Evans, G.D. Farquhar, N.M. Fyllas, P.P.G. Gauthirt, E. Gloor, T.E. Gimeno, K.L. Griffin, R. Guerrieri, M.A. Heskel, C. Huntingford, F.Y. Ishida, J. Kattge, H. Lambers, M.J. Liddell, J. Lloyd, C.H. Lusk, R.E. Martin, A.P. Maksimov, T.C. Maximov, Y. Malhi, B.E. Medlyn, P. Meir, L.M. Mercado, N. Mirotchnik, D. Ng, Ü. Niinemets, O.S. O'Sullivan, O.L. Philips, L. Poorter, P. Poot, I.C. Prentice, N. Salinas, L.M. Rowland, M.G. Ryan, S. Sitch, M. Slot, N.G. Smith, M.H. Turnbull, M.C. VanderWel, F. Valladares, E.J. Veneklaas, L.K. Weerasinghe, C. Wirth, I.J. Wright, K. Wythers, J. Xiang, S. Xiang and J. Zaragoza-Castells (2015) Global variability in leaf respiration in relation to climate, plant functional types and leaf traits. *New Phytologist* **206**: 614–636.

Bernacchi, C.J., E.L. Singsaas, C. Pimentel, A.R. Portis Jr and S.P. Long (2001) Improved temperature response functions for models of Rubisco-limited photosynthesis. *Plant, Cell and Environment* **24**: 253–259.

Campiolo, M. S. Vicca, S. Luyssaert, J. Bilcke, E. Ceschia, F.S. Chapin III, P. Ciais, M. Fernández-Martínez, Y. Malhi, M. Obersteiner, D. Olefeldt, D. Papale, S.L. Piao, J. Peñuelas, P.F. Sullivan, X. Wang, T. Zenonen and I.A. Janssens (2015) Biomass production efficiency controlled by management in temperate and boreal ecosystems. *Nature Geoscience* **8**: 843–846.

Ceccherini, G., N. Gobron, N and M. Robustelli (2013) Harmonization of Fraction of Absorbed Photosynthetically Active Radiation (FAPAR) from Sea-Viewing Wide Field-of-View Sensor (SeaWiFS) and Medium Resolution Imaging Spectrometer Instrument (MERIS). *Remote Sensing* **5**: 3357-3376.

De Kauwe, M.G., T.F. Keenan, B.E. Medlyn, I.C. Prentice and C. Terrer (2016a) Satellite based estimates underestimate the effect of CO_2 fertilization on NPP. *Nature Climate Change* **6**: 892–893.

De Kauwe, M.G., Y.-S. Lin, I.J. Wright, B.E. Medlyn, K.Y. Crous, D.S. Ellsworth, V. Maire, I.C. Prentice, O.K. Atkin, A. Rogers, Ü. Niinemets, S. Serbin, P. Meir, J. Uddling, H.F. Togashi, L. Tarvainen, L.K. Weerasinghe, B.J. Evans, F.Y. Ishida and T. F. Domingues (2016b) A test of a "one-point method" for estimating maximum carboxylation capacity. *New Phytologist* **210**: 1130–1144.

Dee, D.P., S.M. Uppala, A.J. Simmons, P. Berrisford, P. Poli, S. Kobayashi, U. Andrae, M.A. Balmaseda, G. Balsamo, P. Bauer, P. Bechtold, A.C.M. Beljaars, L. van der Berg, J. Bidlot, N. Bormann, C. Delsol, R. Dragani, M. Fuentes, A.J. Geer, L. Haimberger, S.B. Healy, H. Hersbach, E.V. Hólm, L. Isaksen, P. Kållberg, M. Köhler, M. Matricardi, A.P. McNally, B.M. Monge-Sanz, J.-J. Morcrette, B.-K. Park, C. Peubey, P. de Rosnay, C Tavolato, J.-N. Thépaut and F. Vitart (2011) The ERA-Interim reanalysis: configuration and performance of the data assimilation system. *Quarterly Journal of the Royal Meteorological Society* **137**: 553–597.

Dewar, R.C. (1996) The correlation between plant growth and intercepted radiation: an interpretation in terms of optimal plant nitrogen content. *Annals of Botany* **78**: 125–136.

Donohue, R.J., I.H. Hume, M.L. Roderick, T.R. McVicar, J. Beringer, L.B. Hutley, J.C. Gallant, J.M. Austin, E. can Gorsel, J.R. Cleverly, W.S. Meyer and S.K. Arndt (2014) Evaluation of the remote-sensing-based DIFFUSE model for estimating photosynthesis of vegetation. *Remote Sensing of Environment* **155**: 349–365.

Donohue, R.J., M.L. Roderick, T.R. McVicar and G.D. Farquhar (2013) Impact of CO_2 fertilization on maximum foliage cover across the globe's warm, arid environments. *Geophysical Research Letters* **40**: 1–5.

Farquhar, G.D., S. von Caemmerer and J. Berry (1980) A biochemical model of photosynthetic CO_2 assimilation in leaves of C_3 species. *Planta* **149**: 78–90.

Fernández-Martínez, M., S. Vicca, I.A. Janssens, P. Ciais, M. Obersteiner, M. Bartrons, J. Sardans, A. Verger, J.G. Canadell, F. Chevallier, X. Wang, C. Bernhoffer, P.S. Curtis, D. Gianelle, T. Grünwald, B. Heinesch, A. Ibrom, A. Knohl, T. Laurila, B.E. Law, J.M. Limousin, B. Longdoz, D. Loustau, I. Mammarella, G. Matteucci, R.K. Monson, L. Montagnani, E.J. Moors, J.W. Munger, D. Papale, S.L. Piao and J. Peñuelas (2017) Atmospheric deposition, CO_2, and change in the land carbon sink. *Scientific Reports* **7**: 9632.

Garbulsky, M.F., J. Peñuelas, D. Papale, J. Ardö, M.L. Goulden, G. Kiely, A.D. Richardson, E. Rotenberg, E.M. Veenendaal and I. Filella (2010) Patterns and controls of the variability of radiation use efficiency and primary productivity across terrestrial ecosystems. *Global Ecology and Biogeography* **19**: 253–267.

Gill, A.L. and A.C. Finzi (2016) Belowground carbon flux links biogeochemical cycles and resource-use efficiency at the global scale. *Ecology Letters* **19**: 1419–1428.

Ghent, D. (2012) *Land Surface Temperature Validation and Algorithm Verification*. Report to European Space Agency (UL-NILU-ESA-LSTVAV).

Goetz, S.J., S.D. Prince, S.N. Goward, M.M. Thawley and J. Small (1999) Satellite remote sensing of primary production: an improved production efficiency modeling approach. *Ecological Modelling* **122**: 239–255.

Graven, H.D., R.F. Keeling, S.C. Piper, P.K. Patra, B.B. Stephens, S.C. Wofsy, L.R. Welp, C. Sweeney, P.P. Tans, J.J. Kelley, B.C. Daube, E.A. Kort, G.W. Santoni and J.D. Bent (2013) Enhanced seasonal exchange of CO_2 by northern ecosystems since 1960. *Science* **341**: 1085–1089.

Haxeltine, A. and I.C. Prentice (1996) A general model for the light use efficiency of primary production. *Functional Ecology* **10**: 551–561.

Hermida-Carrera, C., M.V. Kapralov and J. Galmés (2016) Rubisco catalytic properties and temperature response in crops. *Plant Physiology* **171**: 2549–2561.

Kattge, J. and W. Knorr (2007) Temperature acclimation in a biochemical model of photosynthesis: a reanalysis of data from 36 species. *Plant Cell and Environment* **30:** 1176–1190.

Keeling, C.D., S.C. Piper, R.B. Bavastow, M. Wahlen, T.P. Whorf, M.R. Raupach and H.A. Meijer (2001) Exchanges of atmospheric CO_2 and $^{13}CO_2$ with the terrestrial biosphere and oceans from 1978 to 2000. I. Global aspects. *SIO Reference Series* **01–06**, Scripps Institution of Oceanography, 88 pp.

Keenan, T.F., I.C. Prentice, J.G. Canadell, C. Williams, H. Wang, M.R. Raupach and G.J. Collatz (2016) Recent pause in the growth rate of atmospheric CO_2 due to enhanced terrestrial uptake. *Nature Communications* **7**: 13428.

Knorr, W. and M. Heimann (1995) Impact of drought stress and other factors on seasonal land biosphere CO_2 exchange studied through an atmospheric tracer transport model. *Tellus B* **47**: 471–489.

Landsberg, J.J. and R.H. Waring (1997) A generalised model of forest productivity using simplified concepts of radiation-use efficiency, carbon balance and partitioning. *Forest Ecology and Management* **95**: 209–228.

Lin, X., B. Chen, J. Chen, H. Zhang, S. Sun, G. Xu, L. Guo, M. Ge, J. Qu, L. Li and Y. Kong (2017) Seasonal fluctuations of photosynthetic parameters for light use efficiency models and the impacts on gross primary production estimation. *Agricultural and Forest Meteorology* **236**: 22–35.

Lin, Y.-S., B.E. Medlyn, R.A. Duursma, I.C. Prentice, H. Wang, D. Eamus, C.V.M. Barton, J. Bennie, D. Bonal, A. Bosc, M.S.J. Broadmeadow, L.A. Cernusak, P. De Angelis, A.C. Lola da Costa, J.E. Drake, D.S. Ellsworth, M. Freeman, O. Ghannoum, T.E. Gimeno, Q. Han, K. Hikosaka, L.B. Hutley, J.W. Kelly, K. Kikuzawa, P. Kolari, K. Koyama, J.-M. Limousin, M.L. Linderson, M. Löw, C. Macinnis-Ng, N.K. Martin-StPaul, P. Meir, T.N. Mikkelsen, P. Mitchell, J.B. Nippert, T.W. Ocheltree, Y. Onoda, M. Op de Beeck, V. Resco de Dios, A. Rey, A. Rogers, L. Rowland, N. Salinas, S.A. Setterfield, W. Sun, L. Tarvainen, S. Tausz-Posch, D.T. Tissue, J. Uddling, G. Wallin, J.M. Warren, L. Wingate and J. Zaragoza-Castells (2015) Optimal stomatal behaviour around the world: synthesis of a global stomatal conductance database. *Nature Climate Change* **5**: 459–464.

Maire, V., P. Martre, J. Kattge, F. Gastal, G. Esser, S. Fontaine and J.-F. Soussana (2012) The coordination of leaf photosynthesis links C and N fluxes in C_3 plant species. *PLOS One* **7**: e38345.

McCallum, I., O. Franklin, E. Moltchanova, L. Merbold, C. Schmullius, A. Shvidenko, D. Schepaschenko and S. Fritz (2013) Improved light and temperature responses for light-use-efficiency-based GPP models. *Biogeosciences* **10**: 6577–6590.

McCallum, I., W. Wagner, C. Schmullius, A. Shvidenko, M. Obersteiner, S. Fritz and S. Nilsson (2009) Satellite-based terrestrial production efficiency modeling. *Carbon Balance and Management* **4**: 8.

Medlyn, B.E. (1998) Physiological basis of the light use efficiency model. *Tree Physiology* **18**: 167–176.

Medlyn, B.E. (2011) Comment on "Drought-Induced Reduction in Global Terrestrial Net Primary Production from 2000 Through 2009". *Science* **333**: 1093.

Medlyn, B.E., R.A. Duursma, D. Eamus, D.S. Ellsworth, I.C. Prentice, C.V.M. Barton, K.Y. Crous, P. De Angelis, M. Freeman and L. Wingate (2011) Reconciling the optimal and empirical approaches to modelling stomatal conductance. *Global Change Biology* **17**: 2134–2144. [Corrigendum: *ibid.* **18**: 3476.]

Meek, D.W., J.L. Hatfield, T.A. Howell, S.B. Idso, S. B. and R.J. Reginato (1984) A generalized relationship between photosynthetically active radiation and solar radiation. *Agronomy Journal* **76**: 939–945.

Michaletz, S.T., D. Cheng, A.J. Kerkhoff and B.J. Enquist (2014) Convergence of terrestrial plant production across global climate gradients. *Nature* **512**: 39–43.

Monteith, J. (1972) Solar radiation and productivity in tropical ecosystems. *Journal of Applied Ecology* **19**: 747–766.

Monteith, J.L. (1977). Climate and the efficiency of crop production in Britain. *Philosophical Transactions of the Royal Society of London* **281**: 277–294.

Ogutu, B.O. and J. Dash (2013) An algorithm to derive the fraction of photosynthetically active radiation absorbed by photosynthetic elements of the canopy ($FAPAR_{ps}$) from eddy covariance flux tower data. *New Phytologist* **197**: 511–523.

Ogutu, B.O., J. Dash and T.P. Dawson (2013) Developing a diagnostic model for estimating terrestrial vegetation gross primary productivity using the photosynthetic quantum yield and Earth Observation data. *Global Change Biology* **19**: 2878–2892.

Potter, C.S., J.T. Randerson, C.B. Field, P.A. Matson, P.M. Vitousek, H.A. Mooney and S.A. Klooster (1993) Terrestrial ecosystem production: a process model based on global satellite and surface data. *Global Biogeochemical Cycles* **7**: 811–841.

Prentice, I.C. (2013) Ecosystem science for a changing world. *Grantham Institute for Climate Change Discussion Paper* **4**, 16 pp.

Prentice, I.C., N. Dong, S.M. Gleason, V. Maire and I.J. Wright (2014) Balancing the costs of carbon gain and water loss: testing a new quantitative framework for plant functional ecology. *Ecology Letters* **17**: 82–91.

Prentice, I.C., X. Liang, B. Medlyn and Y. Wang (2015) Reliable, robust and realistic: the three R's of next-generation land-surface modelling. *Atmospheric Chemistry and Physics* **15**: 5987–6005.

Prince, S.D. and S.N. Goward (1995) Global primary production: a remote sensing approach. *Journal of Biogeography* **22**: 815–835.

Running, S.W. and E.R. Hunt (1993) Generalization of a forest ecosystem process model for other biomes, BIOME-BGC, and an application for global-scale models. In: J.R. Ehleringer and C.B. Field (eds) *Scaling Physiological Processes: Leaf to Globe*, Academic Press, New York, 141–158.

Running, S., F. Nemani, M. Heinsch, M. Zhao, M. Reeves and H. Hashimoto (2004) A continuous satellite-derived measure of global terrestrial primary production. *BioScience* **54**: 547–560.

Ryu, Y., D.D. Baldocchi, H. Kobayashi, C. van Ingen, J. Li, T.A. Black, J. Beringer, E. van Gorsel, A. Knohl, B.E. Law and O. Roupsard (2011) Integration of MODIS land and atmosphere products with a coupled-process model to estimate gross primary productivity and evapotranspiration from 1 km to global scales. *Global Biogeochemical Cycles* **25**: GB4017.

Samanta, A., M.H. Costa, E.L. Nunes, S.A. Vieira, L. Xu and R.B. Myneni (2011) Comment on "Drought-Induced Reduction in Global Terrestrial Net Primary Production from 2000 Through 2009". *Science* **333**: 1093.

Sitch, S., B. Smith, I.C. Prentice, A. Arneth, A. Bondeau, W. Cramer, J.O. Kaplan, S. Levis, W. Lucht, M.T. Sykes, K. Thonicke, S. Venevsky (2003). Evaluation of ecosystem dynamics, plant geography and terrestrial carbon cycling in the LPJ dynamic global vegetation model. *Global Change Biology* **9**: 161–185.

Skillman, J.B. (2008) Quantum yield variation across the three pathways of photosynthesis: not yet out of the dark. *Journal of Experimental Botany* **59**: 1647–1661.

Smith, W.K., S.C. Reed, C.C. Cleveland, A.P. Ballantyne, W.R.L. Anderegg, W.R. Wieder, Y.Y. Liu and S.W. Running (2016) Large divergence of satellite and Earth system model estimates of global terrestrial CO_2 fertilization. *Nature Climate Change* **6**: 306–310.

Stocker, B.D., F. Joos, R. Roth, R. Spahni, L. Bouwman, S. Zaehle, Xu-Ri and I.C. Prentice (2013) Multiple greenhouse gas feedbacks from the land biosphere under future climate change scenarios. *Nature Climate Change* **3**: 666–672.

Stocker, B.D., J. Zscheischler, T.F. Keenan, I.C. Prentice, J. Peñuelas and S.I. Seneviratne (2018) Quantifying soil moisture impacts on light use efficiency across biomes. *New Phytologist*, in press.

Swets, D., B. Reed, J. Rowland and S. Marko (1999) A weighted least-squares approach to temporal NDVI smoothing. In: *Proceedings of the 1999 ASPRS Annual Conference, Portland, Oregon,* pp. 526–536.

Swinnen, E., R. van Hoolst and H. Eerens (2015) *Algorithm Theoretical Basis Document: Dry Matter Productivity (DMP)*. Copernicus, 37 pp.

Tagesson, T., J. Ardö, B. Cappelaere, L. Kergoat, A. Abdi, S. Horion and R. Fensholt (2017) Modelling spatial and temporal dynamics of gross primary production in the Sahel from earth-observation-based photosynthetic capacity and quantum efficiency. *Biogeosciences* **14**: 1333–1348.

Tang, X., H. Li, N. Huang, X. Li, X. Xu, Z. Ding and J. Xie (2015) A comprehensive evaluation of MODIS-derived GPP for forest ecosystems using the site-based FLUXNET database. *Environmental Earth Sciences* **74**: 5907z–5918.

Thomas, R.B., I.C. Prentice, H. Graven, P. Ciais, J.B. Fisher, M. Huang, D.N. Huntzinger, A. Ito, A. Jacobson, A. Jain, J. Mao, A. Michalak, S. Peng, B. Poulter, D.M. Ricciuto, X. Shi, C. Schwalm, H. Tian and N. Zeng (2016) Increased light-use efficiency in northern terrestrial ecosystems indicated by CO_2 and greening observations. *Geophysical Research Letters* **43**: 11339–11349.

Ukkola, A.M., I.C. Prentice, T.F. Keenan, A.I.J.M. van Dijk, N.R. Viney, R.B. Myneni and J. Bi (2015) Reduced streamflow in water-stressed climates consistent with CO_2 effects on vegetation. *Nature Climate Change* **6**: 75-78.

Verma, M., M.A. Friedl, A.D. Richardson, G. Kiely, A. Cescatti, B.E. Law, G. Wohlfahrt, B. Gielen, O. Roupsard, E.J. Moors, P. Toscano, F.P. Vaccari, D. Gianelle, G. Bohrer, A. Varlagin, N. Buchmann, E. van Gorsel, L. Montagnani and P. Propastin (2014) Remote sensing of annual terrestrial gross primary productivity from MODIS: an assessment using the FLUXNET La Thuile data set. *Biogeosciences* **11**: 2185–2200.

Veroustraete, F. (1994) On the use of a simple deciduous forest model for the interpretation of climate change effects at the level of carbon dynamics. *Ecological Modelling* **75**/**76**: 221–237.

Veroustraete, F., H. Sabbe and H. Eerens (2002) Estimation of carbon mass fluxes over Europe using the C-Fix model and Euroflux data. *Remote Sensing of Environment* **83**: 377–400.

Vicca, S., S. Luyssaert, J. Peñuelas, M. Campioli, F.S. Chapin III, P. Ciais, A. Heinemeyer, P. Högberg, W.L. Kutsch, B.E. Law, Y. Malhi, D. Papale, S.L. Piao, M. Reichstein, E.D. Schulze and I.A. Janssens (2012) Fertile forests produce biomass more efficiently. *Ecology Letters* **15**: 520–526.

von Caemmerer, S. (2000) *Biochemical models of leaf photosynthesis*. CSIRO Publishing, Collingwood, 165 pp.

Wang, H., I.C. Prentice, T.W. Davis, T.F. Keenan, I.J. Wright and C. Peng (2016) Photosynthetic responses to altitude: an explanation based on optimality principles. *New Phytologist* **213**: 976–982.

Wang, H., I.C Prentice, W.M. Cornwell, T.F. Keenan, T.W. Davis, I.J. Wright, B.J. Evans and C. Peng (2017) Towards a universal model for carbon dioxide uptake by plants. *Nature Plants* **3**: 734–741.

Wenzel, S., P.M. Cox, V. Eyring and P. Friedlingstein (2014) Emergent constraints on climate-carbon cycle feedbacks in the CMIP5 Earth system models. *Journal of Geophysical Research* **119**: 794–807.

Wenzel, S., P.M. Cox, V. Eyring and P. Friedlingstein (2016) Projected land photosynthesis constrained by changes in the seasonal cycle of atmospheric CO_2. *Nature* **539**: 499–501.

Wright, I.J., P.B. Reich and M. Westoby (2003) Least-cost input mixtures of water and nitrogen for photosynthesis. *The American Naturalist* **161**: 98–111.

Yebra, M., A.I.J.M. van Dijk, R. Leuning and J.P. Guerschman (2015) Global vegetation gross primary production estimation using satellite-derived light-use efficiency and canopy conductance. *Remote Sensing of Environment* **163**: 206–216.

Yuan, W., W. Cai, J. Xia, J. Chen, S. Liu, W. Dong, L. Merbold, B. Law, A. Arain, J. Beringer, C. Bernhofer, A. Black, P.D. Blanken, A. Cescatti, Y. Chen, D. Gianelle, I.A. Janssens, M. Jung, T. Kato, G. Kiely, D. Liu, B. Marcolla, L. Montagnani, A. Raschi, O. Roupsard, A. Varlagin and G. Wohlfahrt (2007) Deriving a light use efficiency model from eddy covariance flux data for predicting daily gross primary production across biomes. *Agricultural and Forest Meteorology* **143**: 189–207.

Yuan, Y., S. Liu, G. Yu, J.-M. Bonnefond, J. Chen K. Davis, A.R. Desai, A.H. Goldstein, D. Gianelle, F. Rossi, A.E. Suyker and S.B. Verma (2010) Global estimates of evapotranspiration and gross primary production based on MODIS and global meteorology data. *Remote Sensing of Environment* **114**: 1416–1431.

Yuan, W.,W. Cai, J. Xia, J. Chen, S. Liu, W. Dong, L. Merbold, B. Law, A. Arain, J. Beringer, C. Bernhofer, A. Black, P.D. Blanken, A. Cescatti, Y. Chen, L. François, D. Gianelle, I.A. Janssens, M. Jung, T. Kato, G. Kiely, D. Liu, B. Marcolla, L. Montagnani, A. Raschi, O. Roupsard, A. Varlagin and G. Wohlfahrt (2014) Global comparison of light use efficiency models for simulating terrestrial vegetation gross primary production based on the LaThuile database. *Agricultural and Forest Meteorology* **192**: 108–120.

Zhao, M., F.A. Heinsch, R.R. Nemani and S.W. Running (2005) Improvements of the MODIS terrestrial gross and net primary production data set. *Remote Sensing of Environment* **95**: 164–176.

Zhao, M. and S.W. Running (2010) Drought-induced reduction in global terrestrial net primary production from 2000 through 2009. *Science* **329**: 940–943.

Zhou, S., R. Duursma, B.E. Medlyn, J.W.G. Kelley and I.C. Prentice (2013) How should we model plant responses to drought? An analysis of stomatal and non-stomatal responses to water stress. *Agricultural and Forest Meteorology* **182–183**, 204–214.

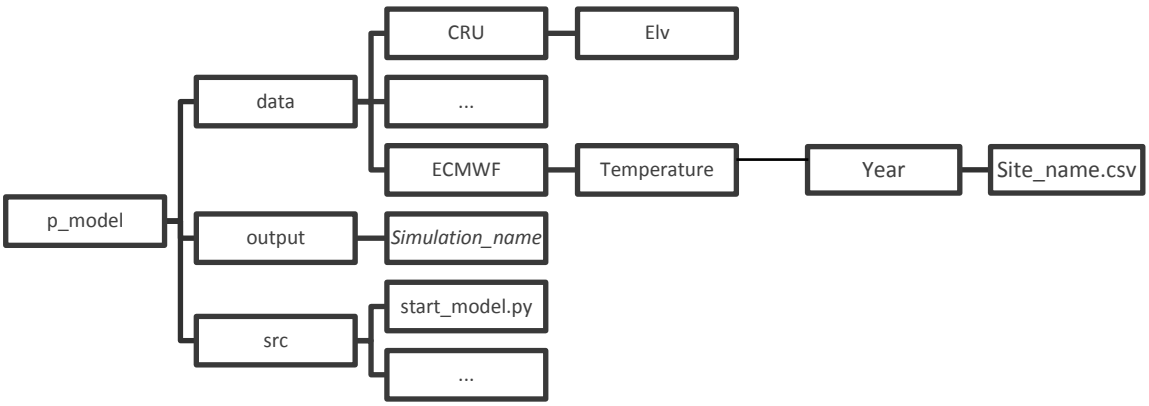
Zhu, X.-G., S.P. Long and D.R. Ort (2010) Improving photosynthetic efficiency for greater yield. *Annual Review of Plant Biology* **61**: 235-261.

Zhu, Z. S. Piao, R.B. Myneni, M. Huang, Z. Zeng, J.G. Canadell, P. Ciais, S. Sitch, P. Friedlingstein, A. Arneth, C. Cao, L. Cheng, E. Kato, C. Koven, Y. Li, X. Lian, Y. Liu, R. Liu, J. Mao, Y. Pan, S. Peng, J. Peñuelas, B. Poulter, T.A.M. Pugh, B.D. Stocker, N. Viovy, X. Wang, Y. Wang, Z. Xiao, H. Yang, S. Zaehle and N. Zeng (2016) Greening of the Earth and its drivers. *Nature Climate Change* **6**: 791–795.

ANNEX A: CODE FOR APPLICATION WITH MERIS GVI AND METEO DATA

Running Terra-P code

To run the code, the driving data must be stored in separate folders within the 'data' folder.



You can specify the data and folder within the 'start_model' script, but all driving data must be stored in the same place. For site-scale runs, you can either have site-scale data, or gridded data stored there. If site data is not found initially, it will be generated from the gridded dataset later on. The diagram shows differtn ways that data can be stored:

1. As individual site data by type, year and then site e.g. ECMWF
2. As one large file e.g. CRU data

The program works as follows:

In the 'start_model' script, the user specifies the location of all the driving data, the variables they wish to save and the name of the simulation (for saving).

The 'main_model' script then checks to see if the data is all at sites scale, or if some is gridded. It also generates the partial differential equation used to calculate uncertainty. It passes all this to the 'run_site' script.

In the 'run_site' script, the sites are looped through and any additional data that needs to be generated for that site is created and saved.

Then the 'run_pmod_site' script is called within the loop which gets the data for the specified site using the 'get_var_data' script. GPP is then calculated for the given data and passed back to the 'run_site' script where it is stored for each site in a dictionary.

The 'main_model' script then saves all the output that is required by the user in a netCDF file by site. All uncertainties are saved with the variables as *var*u.

Start_model.py

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Fri Apr 21 15:31:18 2017

@author: rebeccaThomas
"""


from  main_model_terraP import RUN_P_MODEL
import time
from datetime import datetime
import logging

if __name__ == '__main__':

    # Initialise logging:
    root_logger = logging.getLogger()
    root_logger.setLevel(logging.INFO)


    fh = logging.FileHandler("full_model.log")
    fh_format = logging.Formatter('%(asctime)s - %(lineno)d - %(levelname)-8s: %(funcName)s -
%(message)s')
    fh.setFormatter(fh_format) #, datefmt="%Y-%m-%d %H:%M:%S"
    root_logger.addHandler(fh)

    ch = logging.StreamHandler() #StreamHandler logs to console
    ch.setLevel(logging.WARNING)
    ch_format = logging.Formatter('%(asctime)s -  %(lineno)d - %(levelname)-8s: %(funcName)s -
%(message)s')
    ch.setFormatter(ch_format)
    root_logger.addHandler(ch)

    # This is the simulation name used for saving output
    simsuite = 'ESAcalib'

    start_time = time.clock()
    current_date = datetime.now()
    print current_date
    print "RUNNING P-MODEL..."
    date4saving =  str(current_date.day) + '_' + str(current_date.month)

   # 1. Pick simualtion name

    simName = ('TEST_output_' + simsuite) #Change the simualtion name

    # Only C3 photosynthesis currently tested. In theory, C4 should work
```

```
p_type = 'C3' # Alternative is 'C4'. This effects CO2 and phi_o


# 2. Choose the data you want to use to drive the model. This needs to be
#   saved in a folder of the same name and in the same folder as this code.
# CSV files of the site specific data are not required (as long as there is at least 1)
# as site spcific data is created where this is not given

drivers = {}

                # Data source: extra path to data
drivers['green_driver'] = {'meris':  'csv/SmoothV1'}
drivers['ppfd_driver'] = {'ecmwf' : 'csv/dekadal/Rad'}
drivers['temperature_driver'] = {'ecmwf': 'csv/dekadal/Tavg'} # {'aatsr': 'csv'}
drivers['temperatureMin_driver'] = {'ecmwf': 'csv/dekadal/Tavg'} # {'aatsr': 'csv'}
drivers['temperatureMax_driver'] = {'ecmwf': 'csv/dekadal/Tavg'} # {'aatsr': 'csv'}
drivers['vpd_driver'] = {'ecmwf': 'csv/dekadal/WVP'} #If CRU then vpd will be calulated from CRU vap and avialable temeperature
drivers['co2_driver'] = {'sio_mlo': 'co2'}

# Specify the path to the data (from myhome)
all_data_loc = 'Documents/PhD/terraP_project/data/'
site_info_loc = 'val_site_list/TerraPsites_124_list.csv' # contains csv file of lsit of sites for calcualting GPP
site_data_loc = 'val_site_data/' #If no extra path after data/, then leave this as ''
# Where do you want to save the output?
save_output_loc = 'Documents/PhD/terraP_project/output/val_sites/'


# 3. Choose what output you want.

# What outputs do you want?
# Uncertainty will be saved where given
outr = {}

outr['GPP'] = True
outr['LUE'] = False
outr['TEMP'] = False
outr['SW_IN'] = False
outr['VPD'] = True
outr['fAPAR'] = True
outr['CO2'] = False
outr['GPP_part_u'] = False

# 4. These are the units of the output files: ####
units = {}

units['GPP'] = 'gC m-2 dek-1'
units['LUE'] = 'gC MJ dek-1'
units['TEMP'] = 'celcius'
units['SW_IN'] = 'KJ m-2 dek-1'
```

```
    units['VPD'] = 'hPa'
    units['fAPAR'] = 'n/a'

    # 5. Run the P model

    root_logger.info("Running Pmodel for " + simsuite)

    print 'Running the main model'
    # Initialises the 'main_model' script
    P_MOD = RUN_P_MODEL()
    P_MOD.run_site_mod(drivers,
                       site_info_loc ,
                       site_data_loc ,
                       all_data_loc,
                       save_output_loc,
                       outr,
                       units,
                       simName)



    root_logger.info("FINISHED!")
    end_time = time.clock() - start_time
    print end_time, "seconds"
#
```

Main_model_terraP.py

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Fri Apr 21 16:13:16 2017

@author: rebecca
"""
import os
import numpy as np
import matplotlib.pyplot as plt
from datetime import timedelta, date, datetime
import time
import cPickle as pickle
import logging
import glob

from constants_terraP import C3_phi0
import gpp_uncertainty
from run_site_terraP import RUN_SITE
from make_extra_site_csv import GET_SITE_CSV
from save_station_netcdf import MAKE_NETCDF_SITE

# Myhome is the route directory which all directories are referenced to
```

```
myhome = os.path.expanduser("~")

class RUN_P_MODEL:
    """
    Name: RUN_P_MODEL
    Features: The full P_model
    """
    def __init__(self):
        """
        Name: RUN_P_MODEL.__init__
        Input: - date, start date of run
               - date, end date of run
        """
        # Create a class logger
        self.logger = logging.getLogger(__name__)



    def run_site_mod(self, drivers, site_list_loc, site_data_loc, all_data_loc, save_output_loc, outr, units, simName):

        get_u = True

        site_info_path = os.path.join(myhome, all_data_loc, site_list_loc)
        site_data_path = os.path.join(myhome, all_data_loc, site_data_loc)
        all_data_path = os.path.join(myhome, all_data_loc)
        print "Site_info_path:", site_info_path
        print "Site_data_path:", site_data_path

        # Getting uncertainty equation

        if get_u:
            dif_eq = gpp_uncertainty.get_partial_der()
            get_u = False

        self.logger.info("Getting site data")

        # Figure out if any extra data needs to be got
        data2get, dekads = self.check_data2get(drivers, site_data_path)
        data2get.pop('co2_driver') # remove co2 from the dictionary becuase this is a global value

        # Make CO2 into correct format
        make_co2 = GET_SITE_CSV()
        co2_loc = drivers['co2_driver'].keys()[0]
        make_co2.make_site_co2_csv(dekads, all_data_path, co2_loc, site_data_path)

        self.dekads_out = dekads

        print "Extra data to get:", data2get
        print "DRIVER KEYS:", drivers.keys()
```

```
    # Run script 'run_site' to get site data and run the P-model
    self.site_run = RUN_SITE(outr)
    self.site_run.run_site(drivers, data2get, dekads, site_info_path, site_data_path, all_data_path,
dif_eq, outr) # Run the model for each site in a loop

    self.info_out = self.site_run.site_info
     # Now save the output
    save_output = MAKE_NETCDF_SITE()

    save_this_output = [key for key,val in outr.items() if val==True]

    # this saving may not work- uncomment below if thats the case
    for var2save in save_this_output:
      print 'Saving:', var2save
      if var2save == 'GPP' and outr['GPP_part_u'] is True:
        var2save_u = 'GPP_part_u'
      else:
        var2save_u = var2save + 'u'
      var_info = (var2save, units[var2save])
      var_u_info = (var2save_u, units[var2save])
      filename = os.path.join(myhome, save_output_loc , simName, var2save + '.nc')
      save_output.write_station(filename, self.site_run.__dict__[var2save], self.info_out, dekads,
var_info, \
                    var_in_u = self.site_run.__dict__[var2save_u], var_u_info = var_u_info)



  def read_my_lines(self,csv_reader, lines_list):
    for line_number, row in enumerate(csv_reader):
      print line_number
      print row
      if line_number in lines_list:

        yield line_number, row

  def check_data2get(self, drivers, site_data_path):

    dekads_out = {}
    get_input_data = {}
    start_date = 19000101.0
    end_date = 20171411.0
    for driver_d, source_d in drivers.iteritems():
        # check if datafile exists and read the start and end date if so
        all_these_data_files = os.path.join(site_data_path, (''.join(source_d.keys())),
(''.join(source_d.values())), '*/*.csv')
        print all_these_data_files
        all_files_found = glob.glob(all_these_data_files)

        if all_files_found:
          check_site = os.path.basename(os.path.normpath(all_files_found[0]))[0:6]
          print check_site
```

```
            # Just want to do this for 1 file - so pick the frst station and jsut loop through those years
            these_data_files = os.path.join(site_data_path, (''.join(source_d.keys())),
(''.join(source_d.values())), '*', str(check_site + '*.csv'))
            print these_data_files
            files_found = glob.glob(these_data_files)

            # Then this type of file exists
            self.logger.info("Found data for %s. " %(these_data_files))
            # Now get some info from that to get other files (if needed)
            # Just try the first file for the start and end dates

            for idx in range(len(files_found)):
               all_file = np.genfromtxt(files_found[idx], delimiter = ',', skip_header = 1)
               if idx == 0:
                  all_data_file = all_file
               else:
                  all_data_file = np.concatenate((all_data_file, all_file), axis = 0)


            this_start_date = all_data_file[1][0]
            this_end_date = all_data_file[-1][0]
            # replace start and end dates if they narrow the boundaries
            if this_start_date > start_date:
               start_date = this_start_date
            if this_end_date < end_date:
               end_date = this_end_date

         else:
            # No csv (i.e. site speficic) data found for these files
            # I'm expecting this to happen for any non-ecmwf/MERIS data (i.e. CO2 and VPD)
            self.logger.info("No data file found for %s. Will try to find from other sources"
%(all_these_data_files))
            get_input_data[driver_d] = True


   all_dates = all_data_file[:, 0]
   st_idx = np.where(all_dates == start_date)
   end_idx = np.where(all_dates == end_date)
   print st_idx, end_idx
   all_dekad_times = all_dates[st_idx[0][0] : end_idx[0][0]] # a float of the timespan of driving
data in dekads

   seperate_dates = np.empty([len(all_dekad_times), 3])
#                  seperate_dates = [str(all_dekad_times)[0:4], str(all_dekad_times)[4:6],
str(all_dekad_times)[6:8]]
   for idd in range(len(seperate_dates)):
      seperate_dates[idd] = [str(all_dekad_times[idd])[0:4], str(all_dekad_times[idd])[4:6],
str(all_dekad_times[idd])[6:8]]

   dekads_out['string'] = all_dekad_times
   dekads_out['seperate'] = seperate_dates
```

```
        return get_input_data, dekads_out

    def daterange_daily(self, start_date, end_date):
        for n in range(int ((end_date - start_date).days)):
            yield start_date + timedelta(n)

    def daterange_year(self, start_date, end_date):
            for n in range(int ((end_date - start_date).years)):
                yield start_date + timedelta(n)

    def save2pickle(self, varName, data2save, sim):

        var_sim = (varName + '_'+ sim)
        with open('%s.pickle' % var_sim, 'wb') as out_file:
            pickle.dump(data2save, out_file, pickle.HIGHEST_PROTOCOL)


###### MAIN PROGRAM #########

if __name__ == '__main__':

# This test doens't work - you need to write your own.

    start_time = time.clock()
    current_date = datetime.now()
    print current_date
    print "RUNNING P-MODEL"
    date4saving =  str(current_date.day) + '_' + str(current_date.month)

    sYr = 1998
    sMo = 1
    sDa = 1
    eYr = 2001
    eMo = 12
    eDa = 31

    start_date = date(sYr, sMo, sDa)
    end_date = date(eYr, eMo, eDa)


### site level test
    simsuite = 'AU-Tum_test'
    site_list_loc = 'sofun/input_' + simsuite + '_sofun/site_paramfils/'
    site_data_loc = 'sofun/input_' + simsuite + '_sofun/sitedata/'
    param_data_loc = 'sofun/input_' + simsuite + '_sofun/run/'


    absG = 1.0
    phi_0 = C3_phi0
    beta = 146.0
```

```
    outr = {}
    outr['GPP'] = True
    outr['GPP_u'] = True
    outr['LUE'] = True

    test_gpp =  RUN_P_MODEL('test', 'fluxnet', start_date, end_date)
    test_gpp.run_site_mod(beta, phi_0, absG, outr, site_list_loc, site_data_loc, param_data_loc)

    gpp_out_98 = test_gpp.GPP_out
    gpp_out_unc = test_gpp.GPP_u

    gpp_out = gpp_out_98.get('AU-Tum')[4]
    gpp_out_u = gpp_out_unc.get('AU-Tum')[4]
    gpp_out_t = gpp_out_98.get('AU-Tum')[3]
    plt.plot(gpp_out_t[0:365], gpp_out[0:365])
    plt.fill_between(gpp_out_t[0:365],  gpp_out[0:365]  -  (gpp_out_u[0:365]/2),  gpp_out[0:365]  +
(gpp_out_u[0:365]/2), alpha = 0.2)

    end_time = time.clock() - start_time
    print end_time, "seconds"
```

Run_site_terraP.py

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Fri Apr 21 15:32:12 2017

@author: rebecca
"""

from datetime import timedelta
import numpy as np
import os
import logging
import glob
import csv

from make_extra_site_csv import GET_SITE_CSV
from run_pmod_site_terraP import RUN_PMOD_SITE

myhome = os.path.expanduser("~")

class RUN_SITE:

    def __init__(self, outr):
        """
        Name:  RUN_SITE.__init__
        Input: Site info
        Features: Run the P_model for all sites listed in a file in a loop2
```

```
        """

        self.logger = logging.getLogger(__name__)
        self.logger.info("Initialising P-model at site level")

        self.site_info = {}

        if outr['GPP'] is True:
            self.GPP = {}
            self.GPPu = {}
        if outr['LUE'] is True:
            self.LUE = {}
            self.LUEu = {}
        if outr['TEMP'] is True:
            self.TEMP = {}
            self.TEMPu = {}
        if outr['fAPAR'] is True:
            self.fAPAR = {}
            self.fAPARu = {}
        if outr['SW_IN'] is True:
            self.SW_IN = {}
            self.SW_INu = {}
        if outr['VPD'] is True:
            self.VPD = {}
            self.VPDu = {}
        if outr['CO2'] is True:
            self.CO2 = {}
            self.CO2u = {}

        if outr['GPP_part_u'] is True:
            self.GPP_part_u = {}


    def run_site(self, drivers, data2get, dekads, site_info_path, site_data_path, all_data_path,
dif_eq, outr):
#         # Get site info file

        site_info_file_find = site_info_path
        print site_info_file_find
        site_info_file = glob.glob(site_info_file_find)[0]
        # Read the sitename, start year, end year from the site_info file

        skipped_sites = []

        no_deks = len(dekads['string'])

        with open(site_info_file, 'rb') as csvfile:
            reader = csv.DictReader(csvfile)
            for row in reader:
                run_site = True
                this_site = row['Site ID']
```

```
            this_lat = np.float(row['Lat'])
            this_lon = np.float(row['Lon'])
            # Some files don't have elevation data - if so, need to get it later
            try:
               this_elv = row['Elv']
            except KeyError:
               this_elv = None
            # skip sites with wrong lat/lon
            if abs(this_lat) > 90.0:
               skipped_sites.append(this_site)
               run_site = False
               print 'skipping:', this_site
            if abs(this_lon) > 180.0:
               skipped_sites.append(this_site)
               run_site = False
               print 'skipping:', this_site

            if run_site:
               this_site_id = this_site

               # Getting data for the site
               get_site = GET_SITE_CSV()
               if this_elv is None:
                  #Get elevtion data from CRU if it wasn't provided
                  # Need to have CRU elevation data saved in this repository!
                     data_path_pattern = os.path.join(all_data_path, 'cru', '*elv*.dat')
                     data_path = glob.glob(data_path_pattern)[0]
                     this_elv = get_site.get_cru_elv(this_lat, this_lon, data_path)

               if any(data2get): # If there are any extra data sets to get
               # This is data that isn't provided at site- scale.
               # Need to have gridded data saved in location specified in start script
                  print "Getting extra data for site:", this_site_id
                  self.logger.info("Getting extra data for site: %s" % (this_site_id))

                # Now prepare the data for the remaining variables and add to appropriate files
                  for this_data in data2get.iterkeys():
                     print "Getting extra data:", this_data
                     # Now find the csv file for each data source (needs to have data source in data
folder)
                     data_source = ''.join(drivers[this_data].keys())
                     var_name =  ''.join(drivers[this_data].values())
                     data_path_pattern = os.path.join(all_data_path, data_source, '*%s*.nc'
%(var_name))
                     print data_path_pattern
                     data_path = glob.glob(data_path_pattern)[0]

                     save_loc = os.path.join(site_data_path, data_source, 'csv', var_name)
                     print save_loc
                     save_site = os.path.join(save_loc, str(this_site_id + '.csv'))
```

```
                # Create folder
                if not os.path.exists(os.path.dirname(str(save_loc + '/'))):
                    os.makedirs(os.path.dirname(str(save_loc + '/')))

                var_unit = get_site.make_site_csv(this_lat, this_lon, dekads, data_path, var_name,
save_site)

                info_file_loc = os.path.join(site_data_path, data_source, 'info.txt')
                if not os.path.exists(info_file_loc):
                #Add an info file if it doesnt exists
                  get_site.add_info_file(info_file_loc, data_path, var_name, var_unit)

            self.run_pmod = RUN_PMOD_SITE()
            self.run_pmod.run_pmod(outr, drivers, this_site_id, this_lat, this_lon, this_elv,
site_data_path, dekads, dif_eq)
            self.site_info[this_site_id] = (this_lat, this_lon)

            # Save site output to dictionary to pass back to main_model for saving as netCDF
            if outr['GPP'] is True:
              self.GPP[this_site_id] = self.run_pmod.GPP_out
              self.GPPu[this_site_id] = self.run_pmod.GPP_u
            if outr['LUE'] is True:
              self.LUE[this_site_id] = self.run_pmod.lue_out
              self.LUEu[this_site_id] = self.run_pmod.lue_out_u
            if outr['SW_IN'] is True:
              self.SW_IN[this_site_id] = self.run_pmod.sw_in
              self.SW_INu[this_site_id] = self.run_pmod.sw_in_out_u
            if outr['fAPAR'] is True:
               self.fAPAR[this_site_id] = self.run_pmod.fapar_out
               self.fAPARu[this_site_id] = self.run_pmod.fapar_out_u
            if outr['TEMP'] is True:
               self.TEMP[this_site_id] = self.run_pmod.temp_out
               self.TEMPu[this_site_id] = self.run_pmod.temp_out_u
            if outr['VPD'] is True:
              self.VPD[this_site_id] = self.run_pmod.vpd_out
              self.VPDu[this_site_id] = self.run_pmod.vpd_out_u
            if outr['CO2'] is True:
              self.CO2[this_site_id] = self.run_pmod.co2_out
              self.CO2u[this_site_id] = self.run_pmod.co2_out_u

            if outr['GPP_part_u'] is True:
              self.GPP_part_u[this_site_id] = self.run_pmod.GPP_u_part

    skipped_sites_loc = str(site_info_path + 'skipped_sites.txt')
    print skipped_sites
    np.savetxt(skipped_sites_loc, skipped_sites)
#

  def get_site_basic_info(self, sitename, my_site_info):
```

```
    with open (my_site_info, 'rb') as f:
        for line in f:
            info = line.split()

            try:
                n = info.index('longitude')
                longitude = float(info[n+1])
            except ValueError:
                self.logger.info ("not in this line")

            try:
                n = info.index('latitude')
                latitude = float(info[n+1])
            except ValueError:
                self.logger.info ("not in this line")

            try:
                n = info.index('altitude')
                elv = float(info[n+1])
            except ValueError:
                self.logger.info ("not in this line")

    return longitude, latitude, elv


def get_site_yr_info(self, sitename, param_data_path):

    site_param_file = str(param_data_path + sitename + '.sofun.parameter')
    print site_param_file

    try:
        os.path.isfile(site_param_file)

    except:
            self.logger.debug("Paramter file does not exist for %s" %sitename)
            print "Parameter file does not exist"
#       line_list = []
    with open (site_param_file, 'rb') as f:
        for line in f:
            info = line.split()


            try:
                n = info.index('spinupyears')
                spin_yrs = int(info[n+1])
            except ValueError:
                self.logger.info ("not in this line")

            try:
                n = info.index('recycle')
                recycle_yrs = int(info[n+1])
```

```
            except ValueError:
                self.logger.info ("not in this line")

            try:
                n = info.index('daily_out_startyr')
                start_yr = int(info[n+1])
            except ValueError:
                self.logger.info ("not in this line")

            try:
                n = info.index('daily_out_endyr')
                end_yr = int(info[n+1])
            except ValueError:
                self.logger.info ("not in this line")

        return spin_yrs, recycle_yrs, start_yr, end_yr


    def daterange_daily(self, start_date, end_date):
        for n in range(int ((end_date - start_date).days)):
            yield start_date + timedelta(n)

    def daterange_year(self, start_date, end_date):
            for n in range(int ((end_date - start_date).years)):
                yield start_date + timedelta(n)
```

Run_pmod_site_terraP.py

```
# -*- coding: utf-8 -*-
"""
Created on Fri Apr 21 15:34:48 2017

@author: rebecca
"""

from datetime import date, timedelta, datetime
import numpy as np
import os
import sys
import logging
from constants_terraP import kfFEC

from constants_u import var_u
import gpp_uncertainty

myhome = os.path.expanduser("~")



from gpp_calc_terraP import GPP_CALC
from get_var_data import GET_VAR_DATA
```

```
class RUN_PMOD_SITE:

   def __init__(self):
      """
      Name: RUN_PMOD_SITE.__init__
      Input: none
      """
      self.logger = logging.getLogger(__name__)
      self.logger.info("Running P-model at site level" )

#
   def run_pmod(self, outr, drivers, this_site_id, this_lat, this_lon, site_elv, site_data_path, dekads,
dif_eq):


            var_data = GET_VAR_DATA()
            # Get all data for all years
            data_dict = var_data.get_csv(drivers, dekads, site_data_path, this_site_id)

            make_vpd = True

            print data_dict.keys()
            print make_vpd
            print this_site_id

            site_sw = data_dict['ppfd_driver']
            site_fapar = data_dict['green_driver']
            site_temp = data_dict['temperature_driver']
            site_temp_min = data_dict['temperatureMin_driver']
            site_temp_max = data_dict['temperatureMax_driver']
            site_vpd = data_dict['vpd_driver']
            site_co2 = data_dict['co2_driver']
            site_elv = site_elv

            # Need to convert radiation from Kj/m2/day to mol/m2/day
            # 1. Only get PAR (0.48 * all SW in) (Apparently not needed?)
            # 2. convert units using from flux to energy conversion (µmol/J)
            # 3. Magnitudes (1e3 J --> 1e-6 mol = 1e-3)
            site_ppfd = site_sw * kfFEC * (1.0e-3)

#
            # Now get uncertainty
            # For constant uncertainty, expand var_u to cover entire dekad. For varying uncertainty,
get time varying site spefic uncertainty
            unc_dict = var_data.get_unc_csv(var_u, drivers, dekads, site_data_path, this_site_id)
            unc_dict['rh'] = unc_dict['vpd']

            # run script 'gpp_calc'
            self.make_gpp = GPP_CALC()
```

```
            self.make_gpp.run_grid(site_ppfd, site_fapar, site_temp, site_co2, site_elv,
                                   tMax = site_temp_max, tMin = site_temp_min, site_vpd = site_vpd,
make_vpd = make_vpd, )

            total_gpp_u, partial_u_lue, partial_u_tt, partial_u_fapar, partial_u_vpd, partial_u_ppfd,
partial_u_rest = gpp_uncertainty.calc_uncertainty(dif_eq, \
                                                  unc_dict, site_ppfd, site_fapar, site_temp, site_temp_min,
site_temp_max, site_co2, \
                                                  self.make_gpp.oxy_elv, vpd_in = site_vpd)

            if outr['GPP'] is True:
               self.GPP_out = self.make_gpp.gpp
               self.GPP_u = total_gpp_u

            if outr['LUE'] is True:
               self.lue_out = self.make_gpp.lue
               self.lue_out_u = partial_u_lue
            if outr['TEMP'] is True:
               self.temp_out = site_temp
               self.temp_out_u = unc_dict['temperature']
            if outr['VPD'] is True:
               self.vpd_out = self.make_gpp.vpd_hPa_
               self.vpd_out_u = unc_dict['vpd']
            if outr['fAPAR'] is True:
               self.fapar_out = site_fapar
               self.fapar_out_u = unc_dict['green']
            if outr['SW_IN'] is True:
               self.sw_in = site_sw
               self.sw_in_u = unc_dict['ppfd']
            if outr['CO2'] is True:
               self.co2_out = site_co2
               self.co2_out_u = unc_dict['co2']

            if outr['GPP_part_u'] is True:
               self.GPP_u_part = np.array([total_gpp_u, partial_u_tt, partial_u_fapar,  partial_u_vpd,
partial_u_ppfd, partial_u_rest])


      def daterange_daily(self, start_date, end_date):

         for n in range(int ((end_date - start_date).days)):
            cd = start_date + timedelta(n)
            if cd.month == 2 and cd.day == 29:
               self.logger.info ("leap year")
            else:
               yield cd
```

gpp_calc_terraP.py

```
# -*- coding: utf-8 -*-
"""
```

```
Created on Tue Feb  9 11:47:01 2016

@author: GreencyclesII
"""

######## Import modules#########
import numpy as np
from constants_terraP import c_star, C3_phi0, absG
from m_calc_terraP import M_CALC

###############
# Main program
###############

class GPP_CALC:
    """
    Name: GPP_CALC
    Features: Calculates GPP for given data
    """
    def __init__(self):
        """
        Name:     GPP_CALC.__init__
        Input:    None.
        Features: Initialise class variables

        """
        self.gpp = None
        self.old_frac = None
        self.gpp_old_eq = None

       #       ### Class Function Definitions ####

    def run_grid(self, ppfd, this_fapar, temp, aCO2, elv, tMax = False, tMin = False, site_vpd = False,
make_vpd = False):
        """
        Name: GPP_CALC.run_grid
        Input: - float, incoming light (ppfd) kJ/m2/day --> this gets converted to
               - float, faparn/a
               - float, temperature (temp) C
               - float,   ambient CO2 (aCO2) ppm --> this gets converted to Pa using patm in Mcalc
               - float, elevation (elv) m
               - float, monhtly maximum daily temp (tMax) C
               - float, monthly minimum daily tamperature (tMin) C
               - float, vapour pressure (vap) hPa --> this is used to calucate VPD and is also converted to
Pa

        Outputs: - float, GPP gC/m2/day
                 - float, C13 discrimination
        Depends: - Mcalc

        """
```

```
phi_0 = C3_phi0


my_M=M_CALC()

if make_vpd:
    this_vpd = my_M.calc_vpd(tMax, tMin, site_vpd)
else:
    this_vpd = site_vpd # in hPa

my_M.run_grid(temp, aCO2, elv, this_vpd)
self.M = my_M.m
self.GAMMA_ST = my_M.GSTAR
self.K1_ = my_M.K_1
self.ETA_ST = my_M.ESTAR
self.vpd_hPa_ = this_vpd
self.m_frac = np.sqrt(1.0 - ((c_star / self.M) ** ( 2.0 / 3.0 )))
self.patm = my_M.patm
self.oxy_elv = my_M.oxy_elv


# GPP is zero for monhtly mean temepratures below 0C

self.lue = phi_0 *\
        self.M * \
        self.m_frac

if type(temp) == np.float64:
   if temp < 0.0:
      self.lue = 0.0
else:
    neg_t = np.where(temp < 0.0)
    self.lue[neg_t] = 0.0

self.gpp = self.lue * \
        ppfd * \
        this_fapar  * \
        absG
```

m_calc_terraP.py

```
# -*- coding: utf-8 -*-
"""
Created on Tue Feb  2 16:44:57 2016

@author: RebeccaThomas
```

```
testing the calculation of m
"""
# TO DO:
# 1. Import Ca (CO2), Tk (air temp), Tmax, Tmin, monthly vapour pressure, z (elevation)
# 2. Initialise grid??
# 3. Calculate photorespiratory compensation point (gamma*)
# 4. Calcuate Viscosity of water (eta *)
# 5. Calcualte VPD (D)
# 6. Calcuate K1 --> requires Kc, Po, Ko
# 7. Calculate Kc
# 8. Calulate Po
# 9. Calculate Ko
# 10. When all above calculated--> calculate m
# 11. TO OUTPUT: m only

# Input: Ca (CO2), Tk (air temp), Tmax, Tmin, monthly vapour pressure, z (elevation)
# Output: m



import numpy as np

import os


myhome = os.path.expanduser("~")

from evap import EVAP

from constants_terraP import gamma_25, t_25, Ha, Ha_kc, Ha_ko, R, eta_const, kco, kPo, ko25,
kc25, es0, k2c, beta

class M_CALC:
    """
    Name: M_CALC
    Features: Calculates M for GPP
    """

    def __init__(self):
            """
            Name:     M_CALC.__init__
            Input:    - float, latitude, degrees (lat)
                      - float, latitude, degrees (lon)

            """
            self.m = None
            self.vpd_Pa = None

        #### Class Function dEfinitions ####

    def run_grid(self, temp, aCO2, elv_in,  vpd_in):
```

```
        """
        Name: M_CALC.run
        Input: - float, ambient CO2 (aCO2)
        Outputs: - float, M (for GPP claucation)
        Depends: - GAMMA_STAR
              - ETA_STAR
              - VPD_CALC
              - K1_CALC
        """
        elv = np.asarray(elv_in, dtype=np.float)
        evap = EVAP(25.0, elv)
        self.patm = evap.elv2pres(elv)


        pp_CO2 = aCO2 * 1E-6 * self.patm  # CO2 in Pa--> partial pressure for average atmospheroc
pressure
                                    #

        if type(temp) == np.float64:
           kelv2cel = k2c
        else:
           kelv2cel = np.full_like(temp, k2c)

        k_temp = temp + kelv2cel  #temp in kelvin

        # convert vpd from hPa to Pa
        vpd = vpd_in * 100.0

        m_calc = (pp_CO2 - self.gamma_star(k_temp)) / \
              (pp_CO2 + (2 * self.gamma_star(k_temp)) + \
              (3 * self.gamma_star(k_temp)) * \
              np.sqrt(1.6 * self.eta_star(temp, self.patm, 25.0, kPo, elv) * \
              vpd * \
              (beta ** -1.0) * \
              ((self.k1(k_temp, elv, self.patm) + self.gamma_star(k_temp)) ** -1.0)))


        self.vpd_Pa = vpd
        self.m = m_calc
        self.oxy_elv = self.p0(elv, self.patm)
        return m_calc

    def gamma_star(self,k_temp):
        """
        Name: GAMMA_STAR.run
        Input: -Float, Air temp (Tk) Given in degC, need kelvin
        Output: - Float, Gamma*
        """
        ttg=((k_temp - t_25) * Ha) / (R * k_temp * t_25)
        gs=gamma_25 * np.exp(ttg)
        self.GSTAR = gs
```

```
        return gs

    def eta_star(self,temp, patm, temp25, kPo, elv):
        """
        Name: ETA_STAR.run
        Input: -Float, Air temp (Tk)
        Output: - Float, ETA*
        """

        evap = EVAP(25.0, elv)

        self.ns = evap.viscosity_h2o(temp, patm)
        self.ns25 = evap.viscosity_h2o(temp25, kPo)

        es = self.ns/self.ns25
        self.ESTAR = es
#         print es
        return es

    def calc_vpd(self,tMax, tMin, vap):
        """
        Name: VPD_CALC.run
        Input: -Float, monlty average daily maximum temp (Tmax) deg C
             -Float, monlty average daily minimum temp (Tmin) deg C
             -Float, monlty average vapour pressure (vap) hPa (1 Pa = 0.01 hPa)
             Output: - Float, D (jn equation)
        """

        tMaxMin=(8.635 * (tMax + tMin))/(0.5 * (tMax + tMin) + 237.3)

        vpd_out=(es0 * np.exp(tMaxMin) - (0.10 * vap)) * 10. # kPa --> hPa

        return vpd_out

    def k1(self, k_temp, elv, patm):
        """
        Name:    K0_CALC.__init__
        Input:   - float, latitude, degrees (lat)
               - float, latitude, degrees (lon)
               -Float, monlty average temperature (Tk)
               -Float, elevation at grid square (elv)
               Output: - Float, K1
        Depends: - Kc
              - P0
              - K0
        """
        k1_out = self.kc(k_temp) * (1 + ( self.p0(elv, self.patm) / self.k0(k_temp)))
        self.K_1 = k1_out
        return k1_out
```

```
    def kc(self,k_temp):

        """
        Name: k1_calc.kc
        Input: -Float, monlty average temperature (Tk)
             - Const = 79.430 kJ mol-1 = energy of activation for carboxylation
        Output: - Float, Kc
        """

        tempFrac=((k_temp - t_25) * Ha_kc) / (R * k_temp * t_25)

        Kc_out = kc25 * np.exp(tempFrac)

        return Kc_out

    def p0(self,elv, patm):
        """
        Name: k1_calc.po
        Input: -Float, elevation at grid point
        Output: - Float, P0. O2 partial pressure
        """

        P0_out = kco * (1e-6) * patm

        return P0_out

    def k0(self,k_temp):
        """
        Name: K0_CALC.run
        Input: -Float, monlty average temperature (Tk)
            - Const = 36.380 kJ mol-1 = energy of activation for oxygenation
        Output: - Float, K1
        """


        tempFracK0=((k_temp - t_25) * Ha_ko) / (R * k_temp * t_25)

        K0_out=ko25 * np.exp(tempFracK0)

        return K0_out

    def print_vals(self):

        print " m: %0.10f" % (self.m)
```

gpp_uncertianty.py

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Wed Jun 21 15:14:19 2017
```

```
@author: rebecca
"""

# This is a script to calculate GPP uncertainties to be called from the gpp_calc script using the current variables
# Using sympy
# 1. calculate the partial diferential equations wrt each variable
# 2. Combine these using the stndard formula
# 3. claucate uncertainty for each pixel at each timestep

import sympy as sp
import numpy as np
from constants_terraP import gamma_25, t_25, Ha, Ha_kc, Ha_ko, R, kc25, ko25, k2c, es0, c_star, beta, C3_phi0
#from constants_u import  beta_u, phi0_u, g_star25_u, kc_25_u, ko_25_u, ha_g_u, ha_kc_u, ha_ko_u
# first you need all the symbols required for the equation (all the variables)

#sp.init_printing(use_unicode=True)

def get_partial_der():
    # establish the sybols to use
    phi_0, green, ppfd, co2, temperature, temperatureMin, temperatureMax, rh, g_star_25, ha_g, kc_25, ha_kc, ko_25, ha_ko, oxy_elv, beta_, cstar = \
    sp.symbols('phi_0 green ppfd co2 temperature temperatureMin temperatureMax rh g_star_25 ha_g kc_25 ha_kc ko_25 ha_ko oxy_elv beta_ cstar')

    #beta = 146.0

    # The equations
    g_star = g_star_25 * sp.exp((ha_g/R) * (1 /t_25 - 1/temperature))
    eta_star = sp.exp ( 580 * (1/(temperature - 138)) - (1/160))
    kc = kc_25 * sp.exp((ha_kc / R) * (1 /t_25 - 1/temperature))
    ko = ko_25 * sp.exp((ha_ko / R) * (1 /t_25 - 1/temperature))
    kk = kc *( 1 + (oxy_elv / ko))

    # One estiamte of VPD
#       es = es0 * sp.exp((17.27 * (temperature + kelv2cel)) / ((temperature + kelv2cel) + 237.3)) # Abtew and Melese 2013 - Gepisat documentation
#       ea = (rh / 100) * es
#       vpd = es - ea
    # another estimate of vpd
#       vpd = rh

    vap = rh
    vpd = (es0 * sp.exp(((8.635 * (temperatureMax + temperatureMin))/(0.5 * (temperatureMax + temperatureMin) + 237.3))) - (0.1 * vap)) * 10.0

    m_top = co2 - g_star
```

```
    m_bottom = co2 + (2.0 * g_star) + (3.0 * g_star * sp.sqrt(1.6 * eta_star * vpd * (beta_ **-1.0) *
((kk + g_star)**-1.0)))

    m_out = m_top / m_bottom

    gpp_out = phi_0 * green * ppfd * m_out * sp.sqrt(1 - ((cstar / m_out)**(2.0/3.0)))

    # now partial differentiation wrt each variable


    # I've tried evalf(subs:)
    #           subs
    gp_test = gpp_out.evalf(subs={temperature: 23.0, co2: 1000.0, phi_0: 0.83, green: 0.3, ppfd:
1000.0, rh: 83.0, oxy_elv: 100.0})
#       print gp_test

#       print ('calculated GPP out')

    # Then export it to a numpy function using lambdify
    # If you jsut want the constant unceratinties, them jsut output them

   # if out_der == 'constants':
    dif_gpp_phi0 = sp.diff(gpp_out, phi_0)
    dif_gpp_beta_ = sp.diff(gpp_out, beta_)
    dif_gpp_g_star_25 = sp.diff(gpp_out, g_star_25)
    dif_gpp_kc_25 = sp.diff(gpp_out, kc_25)
    dif_gpp_ko_25 = sp.diff(gpp_out, ko_25)
    dif_gpp_ha_g = sp.diff(gpp_out, ha_g)
    dif_gpp_ha_kc = sp.diff(gpp_out, ha_kc)
    dif_gpp_ha_ko = sp.diff(gpp_out, ha_ko)
    dif_gpp_oxy_elv = sp.diff(gpp_out, oxy_elv)
    dif_gpp_cstar = sp.diff(gpp_out, cstar)

    diff_out_phi0 = sp.utilities.lambdify((phi_0, green, ppfd, co2, temperature, temperatureMin,
temperatureMax, rh, \
    g_star_25, ha_g, kc_25, ha_kc, ko_25, ha_ko, oxy_elv, beta_, cstar), dif_gpp_phi0, 'numpy')

    diff_out_beta = sp.utilities.lambdify((phi_0, green, ppfd, co2, temperature, temperatureMin,
temperatureMax, rh, \
    g_star_25, ha_g, kc_25, ha_kc, ko_25, ha_ko, oxy_elv, beta_, cstar), dif_gpp_beta_, 'numpy')

    diff_out_g_star_25 = sp.utilities.lambdify((phi_0, green, ppfd, co2, temperature,
temperatureMin, temperatureMax, rh, \
    g_star_25, ha_g, kc_25, ha_kc, ko_25, ha_ko, oxy_elv, beta_, cstar), dif_gpp_g_star_25,
'numpy')

    diff_out_kc_25 = sp.utilities.lambdify((phi_0, green, ppfd, co2, temperature, temperatureMin,
temperatureMax, rh, \
    g_star_25, ha_g, kc_25, ha_kc, ko_25, ha_ko, oxy_elv, beta_, cstar), dif_gpp_kc_25, 'numpy')
```

```
    diff_out_ko_25 = sp.utilities.lambdify((phi_0, green, ppfd, co2, temperature, temperatureMin, temperatureMax, rh, \
    g_star_25, ha_g, kc_25, ha_kc, ko_25, ha_ko, oxy_elv, beta_, cstar), dif_gpp_ko_25, 'numpy')

    diff_out_ha_g = sp.utilities.lambdify((phi_0, green, ppfd, co2, temperature, temperatureMin, temperatureMax, rh, \
    g_star_25, ha_g, kc_25, ha_kc, ko_25, ha_ko, oxy_elv, beta_, cstar), dif_gpp_ha_g, 'numpy')

    diff_out_ha_kc = sp.utilities.lambdify((phi_0, green, ppfd, co2, temperature, temperatureMin, temperatureMax, rh, \
    g_star_25, ha_g, kc_25, ha_kc, ko_25, ha_ko, oxy_elv, beta_, cstar), dif_gpp_ha_kc, 'numpy')

    diff_out_ha_ko = sp.utilities.lambdify((phi_0, green, ppfd, co2, temperature, temperatureMin, temperatureMax, rh, \
    g_star_25, ha_g, kc_25, ha_kc, ko_25, ha_ko, oxy_elv, beta_, cstar), dif_gpp_ha_ko, 'numpy')

    diff_out_oxy_elv = sp.utilities.lambdify((phi_0, green, ppfd, co2, temperature, temperatureMin, temperatureMax, rh, \
    g_star_25, ha_g, kc_25, ha_kc, ko_25, ha_ko, oxy_elv, beta_, cstar), dif_gpp_oxy_elv, 'numpy')

    diff_out_cstar = sp.utilities.lambdify((phi_0, green, ppfd, co2, temperature, temperatureMin, temperatureMax, rh, \
    g_star_25, ha_g, kc_25, ha_kc, ko_25, ha_ko, oxy_elv, beta_, cstar), dif_gpp_cstar, 'numpy')

    print ('got diff eq for constants')
    #return diff_out_phi0, diff_out_beta, diff_out_g_star_25, diff_out_ha_g, diff_out_kc_25, diff_out_ha_kc, diff_out_ko_25, diff_out_ha_ko


 #  else: # if you want the time varying uncertainty parts

     # Get the partial differntial wrt each variable
    dif_gpp_temperature = sp.diff(gpp_out, temperature)
    dif_gpp_temperatureMin = sp.diff(gpp_out, temperatureMin)
    dif_gpp_temperatureMax = sp.diff(gpp_out, temperatureMax)
    dif_gpp_green = sp.diff(gpp_out, green)
    dif_gpp_ppfd = sp.diff(gpp_out, ppfd)
    dif_gpp_co2 = sp.diff(gpp_out, co2)
    dif_gpp_rh = sp.diff(gpp_out, rh)

#       print ('got dif')

    diff_out_temperature = sp.utilities.lambdify((phi_0, green, ppfd, co2, temperature, temperatureMin, temperatureMax, rh, \
    g_star_25, ha_g, kc_25, ha_kc, ko_25, ha_ko, oxy_elv, beta_, cstar), dif_gpp_temperature, 'numpy')

#       print 'tt'

    diff_out_temperatureMin = sp.utilities.lambdify((phi_0, green, ppfd, co2, temperature, temperatureMin, temperatureMax, rh, \
```

```
    g_star_25, ha_g, kc_25, ha_kc, ko_25, ha_ko, oxy_elv, beta_, cstar), dif_gpp_temperatureMin, 'numpy')

#       print 'tMin'

    diff_out_temperatureMax = sp.utilities.lambdify((phi_0, green, ppfd, co2, temperature, temperatureMin, temperatureMax, rh, \
    g_star_25, ha_g, kc_25, ha_kc, ko_25, ha_ko, oxy_elv, beta_, cstar), dif_gpp_temperatureMax, 'numpy')

#       print 'tMax'

    diff_out_green = sp.utilities.lambdify((phi_0, green, ppfd, co2, temperature, temperatureMin, temperatureMax, rh, \
    g_star_25, ha_g, kc_25, ha_kc, ko_25, ha_ko, oxy_elv, beta_, cstar), dif_gpp_green, 'numpy')

#       print 'green'

    diff_out_ppfd = sp.utilities.lambdify((phi_0, green, ppfd, co2, temperature, temperatureMin, temperatureMax, rh, \
    g_star_25, ha_g, kc_25, ha_kc, ko_25, ha_ko, oxy_elv, beta_, cstar), dif_gpp_ppfd, 'numpy')

#       print 'ppfd'

    diff_out_co2 = sp.utilities.lambdify((phi_0, green, ppfd, co2, temperature, temperatureMin, temperatureMax, rh, \
    g_star_25, ha_g, kc_25, ha_kc, ko_25, ha_ko, oxy_elv, beta_, cstar), dif_gpp_co2, 'numpy')

#       print 'co2'

    diff_out_rh = sp.utilities.lambdify((phi_0, green, ppfd, co2, temperature, temperatureMin, temperatureMax, rh, \
    g_star_25, ha_g, kc_25, ha_kc, ko_25, ha_ko, oxy_elv, beta_, cstar), dif_gpp_rh, 'numpy')

#       print 'rh'

    print ('got diff eqs for vars')


    # you can then evaluate this for each timestep/pixel
    temperature_part = diff_out_temperature(1.16, 0.3, 1000.0, 1000.0, 23.0, 20.0, 24.0, 83.0, 100.0, 0.83, 0.3, 1000.0, 1000.0, 23.0, 83.0, beta, c_star)
#       print 'temperature_part=', temperature_part

    dif_eq = {}
    dif_eq['temperature'] = diff_out_temperature
    dif_eq['temperatureMin'] = diff_out_temperatureMin
    dif_eq['temperatureMax'] = diff_out_temperatureMax
    dif_eq['green'] =diff_out_green
    dif_eq['ppfd'] =diff_out_ppfd
    dif_eq['co2'] =diff_out_co2
```

```
        dif_eq['vpd'] =diff_out_rh
        dif_eq['phi_0'] =diff_out_phi0
        dif_eq['beta_'] =diff_out_beta
        dif_eq['g_star_25'] =diff_out_g_star_25
        dif_eq['ha_g'] =diff_out_ha_g
        dif_eq['kc_25'] =diff_out_kc_25
        dif_eq['ha_kc'] =diff_out_ha_kc
        dif_eq['ko_25'] =diff_out_ko_25
        dif_eq['ha_ko'] =diff_out_ha_ko
        dif_eq['oxy_elv'] =diff_out_oxy_elv
        dif_eq['cstar'] = diff_out_cstar
        dif_eq['var_order'] = ['phi_0', 'green', 'ppfd', 'co2', 'temperature', 'temperatureMin', 'temperatureMax', 'vpd', 'g_star_25', 'ha_g', 'kc_25', 'ha_kc', 'ko_25', 'ha_ko', 'oxy_elv', 'beta_', 'cstar']

        test_out = 'temperature'
        temperature_out = dif_eq[test_out](1.16, 0.3, 1000.0, 1000.0, 23.0, 20.0, 24.0, 83.0, 100.0, 0.83, 0.3, 1000.0, 1000.0, 23.0, 83.0, beta, c_star)
#       print 'testing_eq=', temperature_out
        return dif_eq
        #diff_out_temperature, diff_out_green, diff_out_ppfd, diff_out_ca, diff_out_rh

      # return diff_out_phi0, diff_out_beta, diff_out_g_star_25, diff_out_ha_g, diff_out_kc_25, diff_out_ha_kc, diff_out_ko_25, diff_out_ha_ko


def calc_uncertainty(dif_eq, var_u, this_ppfd, this_green, this_temp, this_tMin, this_tMax, site_CO2, oxy_elv, vpd_in):
    # variabels are the input variables for that timestep (some will be constants)
    # var_u is a dictionary of the uncertainties for that timestep (some will be constant)

    vars_out = dif_eq.get('var_order')
#    print phi0
#    print 'Uncertainty vars out order', vars_out

    sum_u = np.full_like(this_temp, 0.0) # this is assuming I have a varibale co2lled temp which I will pass in
    partial_u_temperature = np.full_like(this_temp, 0.0)
    partial_u_green = np.full_like(this_temp, 0.0)
    partial_u_lue = np.full_like(this_temp, 0.0)
    partial_u_vpd = np.full_like(this_temp, 0.0)
    partial_u_ppfd = np.full_like(this_temp, 0.0)
    partial_u_rest = np.full_like(this_temp, 0.0)

    # All variables that dont contribute to lue uncertainty:
    lue_unc = ['ppfd', 'green', 'phi_0']

    for var_x in vars_out:
#       print var_x
        dif_wrt = dif_eq[var_x]
```

```
        eval_var = dif_wrt(C3_phi0, this_green, this_ppfd, site_CO2, this_temp, this_tMin, this_tMax, vpd_in, gamma_25, t_25, kc25, Ha_kc, ko25, Ha_ko, oxy_elv, beta, c_star)

        sum_u[np.isnan(eval_var)] = 0.0
        partial_u_temperature[np.isnan(eval_var)] = 0.0
        partial_u_green[np.isnan(eval_var)] = 0.0
        partial_u_lue[np.isnan(eval_var)] = 0.0
        partial_u_ppfd[np.isnan(eval_var)] = 0.0
        partial_u_vpd[np.isnan(eval_var)] = 0.0
        partial_u_rest[np.isnan(eval_var)] = 0.0


        var_unc = (eval_var * var_u[var_x])
#       print var_x, var_u[var_x], var_unc
        sum_u = sum_u + (var_unc**2)

        if var_x not in lue_unc:
            partial_u_lue = partial_u_lue + (var_unc**2)

        # Now for all partial uncertainty
        if var_x == 'temperature':
            partial_u_temperature = np.sqrt(var_unc**2)
        elif var_x == 'green':
            partial_u_green = np.sqrt(var_unc**2)
        elif var_x == 'rh':
            partial_u_vpd = np.sqrt(var_unc**2)
        elif var_x == 'ppfd':
            partial_u_ppfd = np.sqrt(var_unc**2)
        else:
            partial_u_rest = partial_u_rest + (var_unc**2)

    total_u = np.sqrt(sum_u)
    part_u = np.sqrt(partial_u_rest)

    return total_u, partial_u_lue, partial_u_temperature, partial_u_green, partial_u_vpd, partial_u_ppfd, part_u
```

get_var_data.py

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Fri Nov 17 10:49:41 2017

@author: rebecca
"""
# This script reads in the data for a given site and outputs it all to a dictionary to use for driving the model

import numpy as np
import logging
```

```
import glob
import os
from constants_terraP import k2c

class GET_VAR_DATA:

   def __init__(self,):
      """
      Name: RUN_PMOD_SITE.__init__
      Input: none
      """
      self.logger = logging.getLogger(__name__)
      self.logger.info("Getting Var data now to run the model")

   def get_csv(self, drivers, dekads, site_data_path, site_id):

       self.logger.info("Getting data for %s" %site_id )
       site_data = {}

       for driver_d, source_d in drivers.iteritems():
         self.logger.info('getting var data for: %s and %s' % (driver_d, source_d))

         if 'co2_driver' not in driver_d:
            if 'cru' in source_d:
               this_source_loc = 'csv/' + (''.join(source_d.values()))
               print this_source_loc
               these_data_files = os.path.join(site_data_path, (''.join(source_d.keys())),
this_source_loc, str(site_id + '.csv'))
            else:

               these_data_files = os.path.join(site_data_path, (''.join(source_d.keys())),
(''.join(source_d.values())), '*', str(site_id + '*.csv'))
#                print 'These_files:', these_data_files
         else:
            these_data_files = os.path.join(site_data_path, (''.join(source_d.keys())), '*.csv')

         files_found = glob.glob(these_data_files)

         if files_found:
            self.logger.info("Found these files %s" %(files_found))

            site_data[driver_d] = self.read_csv(files_found, dekads['string'], col2read = 1)
         else:
             site_data[driver_d] = np.full_like(dekads['string'], 0.0) # This is if there is no data found
for this file
#          else:
#             these_data_files = os.path.join(site_data_path, (''.join(source_d.keys())), '*.csv')
#             files_found = glob.glob(these_data_files)
#             if files_found:
#                this_data_file = files_found[0]
#                self.logger.info("Found these files %s" %(this_data_file))
```

```
#                 site_data[driver_d] = self.read_csv(this_data_file, dekads['string'])

      # change missing vlaues to nan
#       site_data[site_data == -9999.0 ] = np.nan
       return site_data

   def get_unc_csv(self, var_u, drivers, dekads, site_data_path, site_id):

       self.logger.info("Getting data for %s" %site_id )
       site_data_u = {}

       all_varying_drivers = drivers.keys()

       for var2get in var_u.iterkeys():
#           print 'Getting uncertainty for ', var2get
          var2find = var2get + '_'
          var_driver = [a for a in all_varying_drivers if var2find in a]

          if var_driver:
            driver_d = np.copy(var_driver)
            source_d = drivers[driver_d[0]]

            self.logger.info('getting uncertainty var data for: %s and %s' % (var_driver, source_d))

            if 'co2_driver' not in var_driver:
               if 'cru' in source_d:
                  this_source_loc = 'csv/' + (''.join(source_d.values()))
                  print this_source_loc
                  these_data_files       =       os.path.join(site_data_path,       (''.join(source_d.keys()))),
this_source_loc, str(site_id + '.csv'))
               else:

                  these_data_files       =       os.path.join(site_data_path,       (''.join(source_d.keys()))),
(''.join(source_d.values())), '*', str(site_id + '*.csv'))
            else:
               these_data_files = os.path.join(site_data_path, (''.join(source_d.keys())), '*.csv')

            files_found = glob.glob(these_data_files)
            if files_found:
#                self.logger.info("Found these files %s" %(files_found))
               # assume that uncertainty will come in the same file in a second column
               # For now I only have uncertainty data for MERIS
               ss = np.copy(source_d.keys())
               if  ss in ['meris', 'aatsr'] :
                  self.logger.info("Getting uncertainty from column 2 for %s" %(files_found))
                  site_data_u[var2get] = self.read_csv(files_found, dekads['string'], col2read = 2)
               else:
                  self.logger.info("No uncertainty for ECMWF data for %s" %(files_found))
                  site_data_u[var2get] = self.read_csv(files_found, dekads['string'], col2read = 1) * 0.0 #
Set to no uncertianty for now
            else:
```

```
            site_data_u[var2get] = np.full_like(dekads['string'], 0.0) # this is if there is no data for this site
        else:
          # This assumes that anything not picked up by var_driver is a constant
          one_val = var_u[var2get]
          site_data_u[var2get] = np.full_like(dekads['string'], one_val)

      return site_data_u


  def read_csv(self, files_found, dekads, col2read):
    # Read all columns first to get index of where dekads == data

    if len(files_found) > 1:
    # Now I have all the year seperated. To continue vector GPP calc, concatonate all the files
      for idx in range(len(files_found)):
        all_file = np.genfromtxt(files_found[idx], delimiter = ',', skip_header = 1)
        if idx == 0:
          all_var = all_file
        else:
          all_var = np.concatenate((all_var, all_file), axis = 0)
    else:
      this_data_file = files_found[0]
      all_var = np.genfromtxt(this_data_file, delimiter = ',', skip_header = 1)

    # for the MERIS data, overwrite col2read to be the cetre pixel 'r5_c5'
    if '/meris/' in files_found[0]:
      if col2read != 2: # don't change the column if reading uncertainty
        col2read = 43

    start_idx = np.where(all_var[:,0] == dekads[0])[0][0]
    end_idx = np.where(all_var[:,0] == dekads[-1])[0][0]

    out_var = all_var[start_idx : (end_idx + 1), col2read]


    # Now MERIS data may be nan at the pixel, so if thats the case read the whole area (col 1)

    if '/meris/' in files_found[0]:
      if np.mean(out_var) < -200.0:
        out_var = all_var[start_idx : (end_idx + 1), 1]

     # Set no value to nan for now. Can change to proper missing value later
    out_var[out_var == -9999.0] = np.nan


    # LST comes as kelvin so need to turn this to celcius
    if '/aatsr/' in files_found[0]:
      if col2read !=2:
        out_var = out_var - k2c
```

```
        return out_var
```

save_station_netCDF.py

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Wed Nov 22 11:21:53 2017

@author: rebecca
"""

# Saving the data as a netCDF file using discrete sampling geometry

import netCDF4
import numpy as np
import os


class MAKE_NETCDF_SITE:

    def __init__(self):
     """
     Initialise the netCDF file to be written in a loop
     """

    def write_station(self, filename, var_in, station_meta, dekads, var_info, var_in_u = None, var_u_info = None):

        # First first create the approporiate folder if it doesnt exist

         if not os.path.exists(os.path.dirname(filename)):
            os.mkdir(os.path.dirname(filename))
        # First get some of the data out of the dictionaries

         no_stations = len(var_in.keys())
         station_dekads = dekads['string']

         # Initilise the file
         station_file = netCDF4.Dataset(filename, 'w', format = 'NETCDF4')

    # Must assign featureType as timeSeries
         station_file.featureType = 'timeSeries'

    # Assign the dimensions
         time = station_file.createDimension('time', len (station_dekads))
         station = station_file.createDimension('station', no_stations)
         name_strlen = station_file.createDimension('name_strlen', 1)

    # Creat the variables
         var = station_file.createVariable(var_info[0], np.float64, ('station', 'time',))
```

```
        times = station_file.createVariable('time', np.float64, ('time',))
        station_name = station_file.createVariable('station_ID', str, ('station', 'name_strlen', ))
        latitudes = station_file.createVariable('lat', np.float32, ('station',))
        longitudes = station_file.createVariable('lon', np.float32, ('station',))

        if var_in_u:
            var_u = station_file.createVariable(var_u_info[0], np.float64, ('station', 'time',))

            var_u.coordinates = 'lat lon'
            var_u.units = var_u_info[1] # alter this depending on input
            var_u.description = 'Mean over the dekad'

        # Add some attributes
        var.coordinates = 'lat lon'
        var.units = var_info[1] # alter this depending on input
        var.description = 'Mean over the dekad'

        station_name.cf_role = 'timeseries_id'


        # Now write the data

        # Loop through stations now to get output for each

        for station_idx, station_id in enumerate(var_in.keys()):
#           print station_id
            station_id_char = (np.array([station_id], dtype = 'object'))
#           print station_id_char
            station_name[station_idx] = station_id_char
            var[station_idx] = var_in[station_id]
            if var_in_u:
                var_u[station_idx] = var_in_u[station_id]

            latitudes[station_idx] = station_meta[station_id][0]
            longitudes[station_idx] = station_meta[station_id][1]
            times[:] = station_dekads


        station_file.close()

#%% MAIN PROGRAM
if __name__ == '__main__':

    station_lat = 30.3
    station_lon = 50.5
    station_names = 'TT3456789TT' # Needs to be unique
    station_GPP = {'TT3456789TT': np.array([0, 1, 2, 3])}
    station_dekads = {'string': np.array([20020102., 20020111., 20020121., 20020201.])}
    station_extra = {'TT3456789TT': (station_lat, station_lon)}
    var_info = ('TEST_var', 'test_unit')
    fname = 'test_this.nc'
```

```
    make_netcdf = MAKE_NETCDF_SITE()

    make_netcdf.write_station(fname, station_GPP, station_extra, station_dekads, var_info)
```

evap.py

```
#!/usr/bin/python
#
# evap.py
#
# VERSION: 1.1-dev
# LAST UPDATED: 2016-02-19
#
# ~~~~~~~~~
# license:
# ~~~~~~~~~
# Copyright (C) 2016 Prentice Lab
#
# This file is part of the SPLASH model.
#
# SPLASH is free software: you can redistribute it and/or modify it under
# the terms of the GNU Lesser General Public License as published by
# the Free Software Foundation, either version 2.1 of the License, or
# (at your option) any later version.
#
# SPLASH is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU Lesser General Public License for more details.
#
# You should have received a copy of the GNU Lesser General Public License
# along with SPLASH.  If not, see <http://www.gnu.org/licenses/>.
#
# ~~~~~~~~~~
# citation:
# ~~~~~~~~~~
# T. W. Davis, I. C. Prentice, B. D. Stocker, R. J. Whitley, H. Wang, B. J.
# Evans, A. V. Gallego-Sala, M. T. Sykes, and W. Cramer, Simple process-
# led algorithms for simulating habitats (SPLASH): Robust indices of radiation,
# evapotranspiration and plant-available moisture, Geoscientific Model
# Development, 2015 (in progress)

################################################################################
# IMPORT MODULES:
################################################################################
import logging

import numpy
```

```
#from scipy.stats import nanmean

from const import (kw, kG, kL, kR, kPo, kTo, kMa, kMv, pir)


################################################################################
# CLASSES
################################################################################
class EVAP:
  """
  Name:     EVAP
  Features: This class calculates daily radiation and evapotranspiration
            quantities:
            - PPFD (ppfd_d), mol/m^2/day
            - EET (eet_d), mm/day
            - PET (pet_d), mm/day
            - AET (aet_d), mm/day
            - condensation (cn), mm/day
  Version:  1.0.0-dev
            - replaced radiation methods with SOLAR class [15.12.29]
            - implemented logging [15.12.29]
  """
  # \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
  # Class Initialization
  # ////////////////////////////////////////////////////////////////////////////
  def __init__(self, lat, elv=0.0):
    """
    Name:     EVAP.__init__
    Input:    - float, latitude, degrees (lat)
              - float, elevation, m (elv)
    """
    # Create a class logger
    self.logger = logging.getLogger(__name__)
    self.logger.debug("EVAP class called")

    # Assign default public variables:
    self.elv = elv
    self.logger.debug("elevation set to %f m", elv)


  def sat_slope(self, tc):
    """
    Name:     EVAP.sat_slope
    Input:    float, air temperature (tc), degrees C
    Output:   float, slope of sat vap press temp curve (s)
    Features: Calculates the slope of the sat pressure temp curve, Pa/K
    Ref:      Eq. 13, Allen et al. (1998)
    """
    s = (17.269)*(237.3)*(610.78)*(
       numpy.exp(tc*17.269/(tc + 237.3))/((tc + 237.3)**2)
    )
```

```
    self.logger.debug(
       "calculating temperature dependency at %f degrees", numpy.nanmean(tc))
    return s

def enthalpy_vap(self, tc):
    """
    Name:     EVAP.enthalpy_vap
    Input:    float, air temperature (tc), degrees C
    Output:   float, latent heat of vaporization
    Features: Calculates the enthalpy of vaporization, J/kg
    Ref:      Eq. 8, Henderson-Sellers (1984)
    """
    self.logger.debug(
       "calculating temperature dependency at %f degrees", numpy.nanmean(tc))
    enth_out = (1.91846e6*((tc + 273.15)/(tc + 273.15 - 33.91))**2)
    return enth_out

def elv2pres(self, z):
    """
    Name:     EVAP.elv2pres
    Input:    float, elevation above sea level (z), m
    Output:   float, atmospheric pressure, Pa
    Features: Calculates atm. pressure for a given elevation
    Depends:  Global constants
            - kPo
            - kTo
            - kL
            - kMa
            - kG
            - kR
    Ref:      Allen et al. (1998)
    """
    self.logger.debug("estimating atmospheric pressure at %f m", numpy.nanmean(z))
    p = kPo*(1.0 - kL*z/kTo)**(kG*kMa/(kR*kL))
    #p = kPo*(1.0 - kL*z/kTo)**(kG*kMa/(8.3143*kL))
    return p

def density_h2o(self, tc, p):
    """
    Name:     EVAP.density_h2o
    Input:    - float, air temperature (tc), degrees C
            - float, atmospheric pressure (p), Pa
    Output:   float, density of water, kg/m^3
    Features: Calculates density of water at a given temperature and
            pressure
    Ref:      Chen et al. (1977)
    """
    # self.logger.debug(
    #     ("calculating density of water at temperature %f Celcius and "
    #      "pressure %f Pa") % (tc, p))
```

```
    # Calculate density at 1 atm (kg/m^3):
    po = 0.99983952
    po += (6.788260e-5)*tc
    po += -(9.08659e-6)*tc*tc
    po += (1.022130e-7)*tc*tc*tc
    po += -(1.35439e-9)*tc*tc*tc*tc
    po += (1.471150e-11)*tc*tc*tc*tc*tc
    po += -(1.11663e-13)*tc*tc*tc*tc*tc*tc
    po += (5.044070e-16)*tc*tc*tc*tc*tc*tc*tc
    po += -(1.00659e-18)*tc*tc*tc*tc*tc*tc*tc*tc
    # self.logger.debug("water density at 1 atm calculated as %f kg/m^3", po)

    # Calculate bulk modulus at 1 atm (bar):
    ko = 19652.17
    ko += 148.1830*tc
    ko += -2.29995*tc*tc
    ko += 0.01281*tc*tc*tc
    ko += -(4.91564e-5)*tc*tc*tc*tc
    ko += (1.035530e-7)*tc*tc*tc*tc*tc
    # self.logger.debug("bulk modulus at 1 atm calculated as %f bar", ko)

    # Calculate temperature dependent coefficients:
    ca = 3.26138
    ca += (5.223e-4)*tc
    ca += (1.324e-4)*tc*tc
    ca += -(7.655e-7)*tc*tc*tc
    ca += (8.584e-10)*tc*tc*tc*tc
    # self.logger.debug("temperature coef, Ca, calculated as %f", ca)

    cb = (7.2061e-5)
    cb += -(5.8948e-6)*tc
    cb += (8.69900e-8)*tc*tc
    cb += -(1.0100e-9)*tc*tc*tc
    cb += (4.3220e-12)*tc*tc*tc*tc
    # self.logger.debug("temperature coef, Cb, calculated as %f bar^-1", cb)

    # Convert atmospheric pressure to bar (1 bar = 100000 Pa)
    pbar = (1.0e-5)*p
    # self.logger.debug("atmospheric pressure calculated as %f bar", pbar)

    pw = (ko + ca*pbar + cb*pbar**2.0)
    pw /= (ko + ca*pbar + cb*pbar**2.0 - pbar)
    pw *= (1e3)*po
    return pw

def viscosity_h2o(self, tc, p):
    """
    Name:     LUE.viscosity_h2o
    Input:    - float, ambient temperature (tc), degrees C
              - float, ambient pressure (p), Pa
    Return:   float, viscosity of water (mu), Pa s
```

```
Features: Calculates viscosity of water at a given temperature and
        pressure.
Depends:  density_h2o
Ref:     Huber, M. L., R. A. Perkins, A. Laesecke, D. G. Friend, J. V.
        Sengers, M. J. Assael, ..., K. Miyagawa (2009) New
        international formulation for the viscosity of H2O, J. Phys.
        Chem. Ref. Data, Vol. 38(2), pp. 101-125.
"""
# Define reference temperature, density, and pressure values:
tk_ast = 647.096      # Kelvin
rho_ast = 322.0       # kg/m^3
mu_ast = (1e-6)       # Pa s

# Get the density of water, kg/m^3
rho = self.density_h2o(tc, p)

# Calculate dimensionless parameters:
tbar = (tc + 273.15)/tk_ast
tbarx = tbar**(0.5)
tbar2 = tbar**2
tbar3 = tbar**3
rbar = rho/rho_ast

# Calculate mu0 (Eq. 11 & Table 2, Huber et al., 2009):
mu0 = 1.67752
mu0 += 2.20462/tbar
mu0 += 0.6366564/tbar2
mu0 += -0.241605/tbar3
mu0 = 1e2*tbarx/mu0

# Create Table 3, Huber et al. (2009):
hj0 = (0.520094, 0.0850895, -1.08374, -0.289555, 0., 0.)
hj1 = (0.222531, 0.999115, 1.88797, 1.26613, 0., 0.120573)
hj2 = (-0.281378, -0.906851, -0.772479, -0.489837, -0.257040, 0.)
hj3 = (0.161913,  0.257399, 0., 0., 0., 0.)
hj4 = (-0.0325372, 0., 0., 0.0698452, 0., 0.)
hj5 = (0., 0., 0., 0., 0.00872102, 0.)
hj6 = (0., 0., 0., -0.00435673, 0., -0.000593264)
h = hj0 + hj1 + hj2 + hj3 + hj4 + hj5 + hj6
h_array = numpy.reshape(numpy.array(h), (7, 6))

# Calculate mu1 (Eq. 12 & Table 3, Huber et al., 2009):
mu1 = 0.
ctbar = (1./tbar) - 1.
for i in range(6):
   coef1 = numpy.power(ctbar, i)
   coef2 = 0.
   for j in range(7):
      coef2 += h_array[j][i]*numpy.power((rbar - 1.), j)
   mu1 += coef1*coef2
mu1 = numpy.exp(rbar*mu1)
```

```
    # Calculate mu_bar (Eq. 2, Huber et al., 2009)
    #   assumes mu2 = 1
    mu_bar = mu0*mu1

    # Calculate mu (Eq. 1, Huber et al., 2009)
    mu = mu_bar*mu_ast    # Pa s

    return mu


def psychro(self, tc, p):
    """
    Name:     EVAP.psychro
    Input:    - float, air temperature (tc), degrees C
              - float, atm. pressure (p), Pa
    Output:   float, psychrometric constant, Pa/K
    Features: Calculates the psychrometric constant for a given temperature
              and pressure
    Depends:  Global constants:
              - kMa
              - kMv
    Refs:     Allen et al. (1998); Tsilingiris (2008)
    """
    # self.logger.debug(
    #     ("calculating psychrometric constant at temperature %f Celcius "
    #      "and pressure %f Pa") % (tc, p))

    # Calculate the specific heat capacity of water, J/kg/K
    # Eq. 47, Tsilingiris (2008)


    #t_pos = numpy.where(tc >= 0.0)

    if type(tc) == numpy.float64:
       if tc < 0.0:
          cp = 1.013 * 1e3 #J/kg/K
       else:
          cp = 1.0045714270
          cp += (2.050632750e-3)*tc
          cp += -(1.631537093e-4)*tc*tc
          cp += (6.212300300e-6)*tc*tc*tc
          cp += -(8.830478888e-8)*tc*tc*tc*tc
          cp += (5.071307038e-10)*tc*tc*tc*tc*tc
          cp *= (1e3)

    else:
        t_neg = numpy.where(tc < 0.0)
        cp = 1.0045714270
        cp += (2.050632750e-3)*tc
        cp += -(1.631537093e-4)*tc*tc
```

```
        cp += (6.212300300e-6)*tc*tc*tc
        cp += -(8.830478888e-8)*tc*tc*tc*tc
        cp += (5.071307038e-10)*tc*tc*tc*tc*tc
        cp *= (1e3)

    #Assuming the temeprature dependance of Cp is negligable when t < 0 C (MS revision: Appendix B)
        cp[t_neg] = 1.013 * 1e3 #J/kg/K
    # self.logger.debug("specific heat capacity calculated as %f J/kg/K", cp)

    # Calculate latent heat of vaporization, J/kg
    lv_4_g = self.enthalpy_vap(tc)
    # self.logger.debug(
    #     "enthalpy of vaporization calculated as %f MJ/kg", (1e-6)*lv)

    # Calculate psychrometric constant, Pa/K
    # Eq. 8, Allen et al. (1998)
    g = (cp*kMa*p/(kMv*lv_4_g))
    phyc = g.copy()

    neg_g = numpy.where(g < 0.0)

    #phyc[neg_g] *= 0.0

    return phyc

##############################################################################
# MAIN PROGRAM
##############################################################################
if __name__ == '__main__':
  # Create a root logger:
  root_logger = logging.getLogger()
  root_logger.setLevel(logging.debug)

  # Instantiating logging handler and record format:
  root_handler = logging.FileHandler("evap.log")
  rec_format = "%(asctime)s:%(levelname)s:%(name)s:%(funcName)s:%(message)s"
  formatter = logging.Formatter(rec_format, datefmt="%Y-%m-%d %H:%M:%S")
  root_handler.setFormatter(formatter)

  # Send logging handler to root logger:
  root_logger.addHandler(root_handler)

  # Test one-year of SPLASH:
  my_lat = 37.7
  my_elv = 142.
  my_day = 172
  my_year = 2000
  my_sf = 1.0
  my_temp = 23.0
  my_sw = 0.9
```

```
    my_class = EVAP(my_lat, my_elv)
    my_class.calculate_daily_fluxes(my_sw, my_day, my_year, my_sf, my_temp)
```

constants_terraP.py

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Fri Apr 21 16:14:23 2017

@author: rebecca
"""

alpha = 1.0

gamma_25 = 4.220 # Pa  Gamma* at 25C
t_25 = 298.15 # K 25C in Kelvin
Ha = 37830.0 #J/mol Activiation energy for Gamma*
R = 8.3145 # J/mol/K Universal gas constant
eta_const = -0.0227 # No unit  Constant for viscosity of water relative to its value at 25C (See Wang 2015)
kco =  2.09476e5 #ppm. US standard pressure. (From Beni's code - Ref Bernacchi et al 2001)
kPo = 101325.0  # Standard atmopsheric pressure (Pa), Allen 1973

Ha_ko = 36380 #J/mol  Bernacchi 2001 energy of activation for oxygenation
Ha_kc = 79430 # J/mol Bernacchi 2001 energy of activation for carboxylation

kc25 = 39.97 # Pa at 25 deg C and 98.716KPa
ko25 = 27480 # Pa, at 25 deg C and 98.716KPa
es0 = 0.611

beta = 160.0 # Email from Wang Han # No unit  Ratio of carboxylation and transpiration costs at 25C--> for ci:ca calcualtions (least cost hypothesis. Using approximate value = 240

c_star = 0.406 # unit of carbon cost for maintenance of electron transport capacity (obs Jmax:Vc max) (Wang Han emails)
a_hat = 4.4 # Standard value for c13 discrimination - diffusion component (Wang 2015 Eq s42)
b_hat = 27.0 # Standard value for c13 discrimination - biochemical component (Wang 2015 Eq s42)

c_molmass =  12.0107  # g C / mol C

C3_phi0 = 0.092 * c_molmass # # mol/mol From Skillman 2008 & Long et al. 1993 # gC/mol Intrinisic quantum yield of photosynthesis for C3 plants
C4_phi0 = 0.055 * c_molmass # # mol/mol From Skillman 2008 & Long et al. 1993 # gC/mol Intrinisic quantum yield of photosynthesis for C4 plants

absG = 1.0 # Muliptying facotr for fAPAR sometimes
kfFEC = 2.04     # from flux to energy conversion, umol/J (Meek et al., 1984) # From Tyler/Beni

k2c = 273.15
```

constants_u.py

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Fri Jun 23 12:11:43 2017

@author: rebecca
"""

var_u = {}
# These may change. Unertainty values are based on most simialr experiments
var_u['beta_'] = 2.73
var_u['phi_0'] =  0.0
var_u['g_star_25'] = 0.1 #0.12
var_u['kc_25']=   3.45 #1.99
var_u['ko_25'] =   0.0 #8872.74
var_u['ha_g'] =   5020.93 #4692.43
var_u['ha_kc'] =   5018.5 #4989.78
var_u['ha_ko'] =   152.74 #144.0
var_u['oxy_elv'] =  0.0
var_u['cstar'] = 0.112 # This is responisble for a large part of the uncertainty

# Keep these values like this - they will get overwritten by actual data when they are called
var_u['temperature'] = 0.0
var_u['temperatureMin'] = 0.0
var_u['temperatureMax'] = 0.0
var_u['vpd'] = 0.0
var_u['green'] = 0.0
var_u['ppfd'] = 0.0
var_u['co2'] = 0.0
```

const.py

```
# GLOBAL CONSTANTS:
################################################################################
kA = 107.       # constant for Rnl (Monteith & Unsworth, 1990)
kalb_sw = 0.17   # shortwave albedo (Federer, 1968)
kalb_vis = 0.03  # visible light albedo (Sellers, 1985)
kb = 0.20       # constant for Rnl (Linacre, 1968)
kc = 0.25       # cloudy transmittivity (Linacre, 1968)
kCw = 1.05      # supply constant, mm/hr (Federer, 1982)
kd = 0.50       # angular coefficient of transmittivity (Linacre, 1968)
ke = 0.0167     # eccentricity for 2000 CE (Berger, 1978)
keps = 23.44     # obliquity for 2000 CE, degrees (Berger, 1978)
kfFEC = 2.04     # from flux to energy conversion, umol/J (Meek et al., 1984)
kG = 9.80665     # gravitational acceleration, m/s^2 (Allen, 1973)
kGsc = 1360.8    # solar constant, W/m^2 (Kopp & Lean, 2011)
kL = 0.0065      # temperature lapse rate, K/m (Allen, 1973)
kMa = 0.028963   # molecular weight of dry air, kg/mol (Tsilingiris, 2008)
```

```
kMv = 0.01802    # molecular weight of water vapor, kg/mol (Tsilingiris, 2008)
kPo = 101325     # standard atmosphere, Pa (Allen, 1973)
kR = 8.31447     # universal gas constant, J/mol/K (Moldover et al., 1988)
kTo = 298.15     # base temperature, K (Berberan-Santos et al., 1997)
kWm = 150.       # soil moisture capacity, mm (Cramer & Prentice, 1988)
kw = 0.26        # entrainment factor (Lhomme, 1997; Priestley & Taylor, 1972)
komega = 283.0   # longitude of perihelion for 2000 CE, degrees (Berger, 1978)
pir = (numpy.pi/180.0)
```