Version 1.0

# Table of Contents

Photosynthesis Module Overview 1

Introduction 1

A flow chart of the model 2

Model Documentation 3

Description of Modules 3

Interface 3

Namespace Index 4

Class Index 4

File Index 4

photomod 4

Class Documentation 5

photomod::CGasExchange 5

tparms 25

photomod::CGasExchange::tParms 26

File Documentation 29

gas\_exchange.cpp 29

gas\_exchange.h 31

Photosynthesis(notManaged).cpp 32

stdafx.cpp 36

stdafx.h 36

targetver.h 37

Index 37

# Photosynthesis Module Overview

## Introduction

This is a coupled model of leaf gas-exchange and photosynthesis-stomatal conductance-energy balance for a maize leaf. The C++ GasExchange class encapsulates all the calculations to estimate CO2 assimilation rate, stomatal conductance, leaf temperature, and transpiration for a square meter of leaf. The calculations are based on von Caemmerer(2000) C4 model, Kim and Leith (2003) C3 rose model, BWB stomatal conductance(Ball et al., 1987) and energy balance model as described in Campbell and Norman (1998)

##### Authors:

Soo-Hyung Kim, Univ. of Washington

Dennis Timlin, USDA-ARS

David Fleisher, USDA-ARS

##### Version:

1.0

##### Date:

June 2016

**License:** This project is released under the GNU Public License

**Bibliography**

* Ball J.T., Woodrow I.E., Berry J.A. 1987. A model predicting stomatal conductance and its contribution to the control of photosynthesis under different environmental conditions. In: Biggens J, ed. Progress in photosynthesis research. The Netherlands: Martinus Nijhoff Publishers.
* Campbell, G.S., and J.M. Norman. 1998. The light environment of plant canopies. An Introduction to Environmental Biophysics. Springer, New York. pp: 247-281.
* Kim, S.-H., and J.H. Lieth. 2003. A coupled model of photosynthesis, stomatal conductance and transpiration for a rose leaf (Rosa hybrida L.). Ann. Bot. 91:771-781.
* Kim, S.-H., D.C. Gitz, R.C. Sicher, J.T. Baker, D.J. Timlin, and V.R. Reddy. 2007. Temperature dependence of growth, development, and photosynthesis in maize under elevated CO2. Env. Exp. Bot. 61:224-236.
* Kim, S.-H., R.C. Sicher, H. Bae, D.C. Gitz, J.T. Baker, D.J. Timlin, and V.R. Reddy. 2006. Canopy photosynthesis, evapotranspiration, leaf nitrogen, and transcription
* von Caemmerer, S., 2000. Biochemical models of leaf photosynthesis. CSIRO Publishing, Collingwood, Australia.

**Source Code**

GitHub https://github.com/ARS-CSGCL-DT

# A flow chart of the model

# Model Documentation

## Description of Modules

These pages describe the model in detail

# Interface

**Program** \_tmain

##### Interface to photosynthesis module

This program demonstrates how to call the photosynthesis module

##### Use of this interface

Two input files are needed (**parameters.csv** and **ClimateIn.dat** ) and one output file is created (**Results.dat** ). ClimateIn.dat is the default name, any file can be input when prompted by the program. A detailed description of the input files follows.

1. A comma delimited parameter file containing the parameters for the photosynthesis module, one line for each species (Parameters.csv)

2. An comma delimted environmental file (ClimateIn.dat) each line should have these variables (separated by commas):

* temperature (C)
* PAR (umol photons m-2 s-1)
* CO2 content (umol mol-1)
* humidity (%)
* wind (m s-1)
* a flag (0,1) to tell the program if constant temperature is used (or let temperature of leaf vary with stomatal conductance).

3. an output file is produced with results (Results.dat) each line is written as:

* PAR (umol photons m-2 s-1)
* Net Photosynthesis (umol CO2 m-2 s-1)
* Gross Photosynthesis
* VPD (kPa)
* LeafTemperature (C)
* BoundaryLayerConductance (mol m-2 s-1)
* Internal CO2 (umol mol-1)
* Respiration (umolCO2 m-2 s-1)
* Transpiration (umol H2O m-2 s-1)

4. In your calling program, execute the function SetParams first to initialize the calculator for a specific plant species Then execute the function: **SetVal(PFD, Temperature, CO2, RelativeHumidity, Wind, Pressure, ConstantTemperature)** to pass environmental parameters needed to calculate carbon assimilation and transpiration. This will call the function **GasEx()** which carries out the calculations

5. Use the public get functions described in the CGasExchange Class to retrieve the calculated variables.

# Namespace Index

## Namespace List

Here is a list of all namespaces with brief descriptions:

**photomod**

# Class Index

## Class List

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

**photomod::CGasExchange (Class for gas exchange calculations**

**This class simulates gas exchange in plant leaves for C3 and C4 plants )**

**tparms (Structure to hold plant parameters for the photosynthesis model )**

**photomod::CGasExchange::tParms (Parameter description )**

# File Index

## File List

Here is a list of all files with brief descriptions:

**gas\_exchange.cpp (Contains code to simulate gas exchange )**

**gas\_exchange.h (Function headers for calculations )**

**Photosynthesis(notManaged).cpp (Defines the entry point for the console application )**

**stdafx.cpp**

**stdafx.h**

**targetver.h**

# Namespace Documentation

## photomod Namespace Reference

### Classes

* class **CGasExchange**

*Class for gas exchange calculations*

*This class simulates gas exchange in plant leaves for C3 and C4 plants.*

### Detailed Description

photomod is the namespace and contains the code needed to run the model. In includes an interface **\_tmain** (named by Visual Studio), and **gasexchange.cpp** , the model itself

# Class Documentation

## photomod::CGasExchange Class Reference

Class for gas exchange calculations

This class simulates gas exchange in plant leaves for C3 and C4 plants.

#include <gas\_exchange.h>

### Classes

* struct **tParms**

### *parameter description* Public Member Functions

* **CGasExchange** ()

*This is the constructor.*

* **~CGasExchange** (void)

*This is the destructor.*

* double **get\_ANet** ()

*return net photosynthesis (umol CO2 m-2 s-1)*

* double **get\_AGross** ()

*return gross photosynthesis (umol CO2 m-2 s-1)*

* double **get\_Transpiration** ()

*return transpiration rate (umol H2O m-2 s-1)*

* double **get\_LeafTemperature** ()

*return leaf temperature (C)*

* double **get\_Ci** ()

*return internal CO2 concentration (umol mol-1)*

* double **get\_StomatalConductance** ()

*return stomatal conductance to water vapor (mol m-2 s-1)*

* double **get\_BoundaryLayerConductance** ()

*return boundary layer conductance (mol m-2 s-1)*

* double **get\_Respiration** ()

*return respiration rate (umol CO2 m-2 s-1)*

* double **get\_VPD** ()

*return vapor pressure deficit (kpa)*

* void **SetParams** (**tParms** \***sParms**)

*used to pass structure with parameters from the interface to the calculator*

* void **SetVal** (double **PhotoFluxDensity**, double **Tair**, double **CO2**, double **RH**, double **wind**, double **Press**, bool ConstantTemperature)

*sets input values for calculations for a particular set of environmental variables*

### Public Attributes

* struct **photomod::CGasExchange::tParms** **sParms**

*Variable of type* ***tParms****.*

### Protected Member Functions

* void **GasEx** ()

*Main module to calculate gas exchange rates.*

* void **PhotosynthesisC3** (double **Ci**)

*Function to calculate C3 photosynthesis.*

* void **PhotosynthesisC4** (double **Ci**)

*Function to calculate C4 photosynthesis.*

* void **EnergyBalance** ()

*calculates leaf temperature and transpiration*

* double **SearchCi** (double CO2i)

*called iterively to find optimal internal CO2 concentration returns optimal internal CO2 concentration (CO2i)*

* double **EvalCi** (double **Ci**)

*Calls photosynthesis modules to evaluate Ci dependent calculations returns difference between old Ci and new Ci.*

* double **CalcStomatalConductance** ()

*Stomatal conductance (mol m-2 s-1)*

* double **CalcTurbulentVaporConductance** ()

*Conductance for turbulant vapor transfer in air - forced convection (mol m-2 s-1)*

* double **Es** (double Temperature)

*Saturated vapor pressure at given temperature.*

* double **Slope** (double Temperature)

*Slope of the vapor pressure curve.*

* double **QuadSolnUpper** (double a, double b, double c)

*Upper part of quadratic equation solution.*

* double **QuadSolnLower** (double a, double b, double c)

*Lower part of quadratic equation solution.*

* double **minh** (double fn1, double fn2, double theta2)

*Hyperbolic minimum.*

* double **get\_CiCaRatio** ()

*Ratio between internal and atmospheric CO2.*

* double **Square** (double a)

*Squares number.*

* double **Min** (double a, double b, double c)

*Finds minimum of three numbers.*

### Protected Attributes

* double **PhotoFluxDensity**

*Photosynthetic Flux Density (umol photons m-2 s-1.*

* double **R\_abs**

*Absorbed incident radiation (watts m-2)*

* double **Tair**

*Air temperature at 2m, (C)*

* double **CO2**

*CO2 concentration (umol mol-1 air)*

* double **RH**

*Relative Humidity (%, i.e., 80)*

* double **wind**

*Windspeed at 2 meters (km s-1)*

* double **width**

*Leaf width (m)*

* double **Press**

*Air pressure (kPa)*

* double **AssimilationNet**

*Net photosynthesis (umol CO2 m-2 s-1)*

* double **AssimilationGross**

*Gross photosynthesis (umol CO2 m-2 s-1) (Adjusted for respiration)*

* double **Transpiration**

*Transpiration mol H2O m-2 s-1.*

* double **Tleaf**

*Leaf temperature C.*

* double **Ci**

*Internal CO2 concentration umol mol-1.*

* double **StomatalConductance**

*Stomatal conductance umol m-2 s-1.*

* double **BoundaryLayerConductance**

*Boundary layer conductance umol m-2 s-1.*

* double **DarkRespiration**

*Plant respiration umol m-2 s-1.*

* double **VPD**

*Vapor Pressure Density, kPa \*/.*

* double **Ci\_Ca**

*Ratio of internal to external CO2, unitless.*

* double **errTolerance**

*Error tolerance for iterations.*

* double **eqlTolerance**

*Equality tolerance.*

* int **iter\_total**

*Total number of iterations.*

* \*std::string **PlantType**

*For C3 or C4 designation \*/.*

* bool **ConstantLeafTemperature**

*if true, uses constant temperature - if true, does not solve for leaf temperature \*/*

* int **iter1**
* int **iter2**

*Iteration counters.*

* int **iter\_Ci**

*Iteration value for Ci umol mol-1, internal CO2 concentration.*

* bool **isCiConverged**

*True if Ci (internal CO2 concentration) iterations have converged.*

### Detailed Description

Class for gas exchange calculations

This class simulates gas exchange in plant leaves for C3 and C4 plants.

##### Usage

* Use **SetParams** to initialize the GasExchange object with parameters for a specific variety of plant
* Use **SetVal** to pass environmental variables for a simulation and return a structure with output.
* See **Interface** for details on input and output

Definition at line 88 of file gas\_exchange.h.

### Constructor & Destructor Documentation

#### photomod::CGasExchange::CGasExchange ()

This is the constructor.

Constructor - initialization of some variables is done here.

Definition at line 37 of file gas\_exchange.cpp.

43 {

44

45

46 isCiConverged=false;

47 errTolerance = 0.001;

48 eqlTolerance = 1.0e-6;

49

50 }

#### photomod::CGasExchange::~CGasExchange (void )

This is the destructor.

Destructor - nothing is done here

Definition at line 53 of file gas\_exchange.cpp.

57 {

58 }

### Member Function Documentation

#### double photomod::CGasExchange::CalcStomatalConductance ()[protected]

Stomatal conductance (mol m-2 s-1)

calculates and returns stomatal conductance for CO2 in umol CO2 m-2 s-1 Uses Ball-Berry model.

##### See also:

**Es()**

##### Returns:

stomatal conductance for CO2 umol m-2 s-1

**Ds** , VPD at leaf surface

**aa** , a value in quadratic equation

**bb** , b value in quadratic equation

**cc** , calcuation variable (x) in quadratic equation

**hs** , solution for relative humidity

**Cs** , estimate of mole fraction of CO2 at the leaf surface

**Gamma** , CO2 compensation point in the absence of mitochondirial respiration, in ubar

**StomatalConductance** , temporary variable to hold stomatal conductance

Definition at line 394 of file gas\_exchange.cpp.

404 {

405 //\*\* \code

406 double Ds,

407 aa,

408 bb,

409 cc,

410 hs,

411 Cs,

412 Gamma,

413 StomatalConductance;

414 Gamma = 10.0;

415 //\*\* \endcode

416

417 double P=Press/100;

418 Cs = (CO2 - (1.37\*AssimilationNet/BoundaryLayerConductance))\*P; // surface CO2 in mole fraction

419 if (Cs == Gamma) Cs = Gamma + 1;

420 if (Cs <= Gamma) Cs = Gamma + 1;

421 // Quadratic equation to obtain hs by combining StomatalConductance with diffusion equation

422 aa = sParms.g1\*AssimilationNet/Cs;

423 bb = sParms.g0+BoundaryLayerConductance-(sParms.g1\*AssimilationNet/Cs);

424 cc = (-RH\*BoundaryLayerConductance)-sParms.g0;

425 hs = QuadSolnUpper(aa,bb,cc);

426 if (hs > 1) hs = 1;

427 if (hs<0) hs = 0;

428 Ds = (1-hs)\*Es(Tleaf); // VPD at leaf surface

429 StomatalConductance = (sParms.g0+sParms.g1\*(AssimilationNet\*hs/Cs));

430 if (StomatalConductance < sParms.g0) StomatalConductance=sParms.g0; //Limit StomatalConductance to mesophyll conductance

431 return StomatalConductance; // moles m-2 s-1

432 }

#### double photomod::CGasExchange::CalcTurbulentVaporConductance (void )[protected]

Conductance for turbulant vapor transfer in air - forced convection (mol m-2 s-1)

calculates conductance for turbulant vapor transfer in air - forced convection

##### Returns:

conductance for turbulent vapor transfer in air (mol m-2 s-1)

< temporary holding variable for stomatal ratio calculations

< characteristic dimension of leaf

Definition at line 438 of file gas\_exchange.cpp.

439 {

445 double ratio;

446 double Char\_Dim;

447 ratio = Square(sParms.stomaRatio+1)/(Square(sParms.stomaRatio)+1);

448 Char\_Dim = sParms.LfWidth\*0.72; // characteristic dimension of a leaf, leaf width in m

449 // wind is in m per second

450 return (1.4\*0.147\*sqrt(\_\_max(0.1,wind)/Char\_Dim))\*ratio;

451 // multiply by 1.4 for outdoor condition, Campbell and Norman (1998), p109, gva

452 // multiply by ratio to get the effective blc (per projected area basis), licor 6400 manual p 1-9

453 }

#### void photomod::CGasExchange::EnergyBalance ()[protected]

calculates leaf temperature and transpiration

Calculates Transpiration rate (T) and leaf temperature (Tleaf).

Iterates by recalculating photosynthesis until leaf temperatures converge

See Campbell and Norman (1998) pp 224-225

Because Stefan-Boltzman constant is for unit surface area by denifition, all terms including sbc are multilplied by 2 (i.e., RadiativeConductance, thermal radiation)

Does not have input

##### Returns:

nothing but calculates transpiration (T) and leaf temperature (Tleaf)

Definition at line 324 of file gas\_exchange.cpp.

341 {

342 const long Lambda = 44000; //latent heat of vaporization of water J mol-1 - not used in this implementation

343 const double Cp = 29.3; // thermodynamic psychrometer constant and specific hear of air, J mol-1 C-1

344 const double psc = 6.66e-4; //psycrometric constant units are C-1

345 //psc=Cp/Lambda = 29.3/44000 See Campbell and Norman, pg 232, after eq 14.11

346

347 //The following are secondary variables used in the energy balance

348 double HeatConductance, //heat conductance J m-2 s-1

349 VaporConductance, //vapor conductance ratio of stomatal and heat conductance mol m-2 s-1

350 RadiativeConductance, //radiative conductance J m-2 s-1

351 RadiativeAndHeatConductance, //radiative+heat conductance

352 psc1, // apparent psychrometer constant Campbell and Norman, page 232 after eq 14.11

353 Ea, //ambient vapor pressure kPa

354 thermal\_air; // emitted thermal radiation Watts m-2

355 double lastTi, newTi;

356 int iter;

357

358 HeatConductance = BoundaryLayerConductance\*(0.135/0.147); // heat conductance, HeatConductance = 1.4\*.135\*sqrt(u/d), u is the wind speed in m/s} Boundary Layer Conductance to Heat

359 // Since BoundaryLayerConductance is .147\*sqrt(u/d) this scales to 0.135\*sqrt(u/d) - HeatConductance on page 109 of Campbell and Norman, 1998

360 // Wind was accounted for in BoundaryLayerConductance already as BoundaryLayerConductance (turbulent vapor transfer) was calculated from CalcTurbulentVaporConductance() in GasEx.

361 // units are J m-2 s-1

362 VaporConductance = StomatalConductance\*BoundaryLayerConductance/(StomatalConductance+BoundaryLayerConductance); //vapor conductance, StomatalConductance is stomatal conductance and is given as gvs in Campbell and Norman.

363 // note units are moles m-2 s-1.

364 RadiativeConductance = (4\*epsilon\*sbc\*pow(273+Tair,3)/Cp)\*2; // radiative conductance, \*2 account for both sides

365 RadiativeAndHeatConductance = HeatConductance + RadiativeConductance;

366 thermal\_air = epsilon\*sbc\*pow(Tair+273,4)\*2; //Multiply by 2 for both surfaces

367 psc1 = psc\*RadiativeAndHeatConductance/VaporConductance;

368 this->VPD = Es(Tair)\*(1-RH); // vapor pressure deficit Es is saturation vapor pressure at air temperature

369 // iterative version

370 newTi=-10;

371 iter=0;

372 lastTi=Tleaf;

373 double Res, dRes; //temporary variables

374 double thermal\_leaf;

375 Ea = Es(Tair)\*RH; // ambient vapor pressure

376 while ((abs(lastTi-newTi)>0.001) && (iter <maxiter))

377 {

378 lastTi=newTi;

379 Tleaf= Tair + (R\_abs- thermal\_air-Lambda\*VaporConductance\*this->VPD/Press)/(Cp\*RadiativeAndHeatConductance+Lambda\*Slope(Tair)\*VaporConductance); // eqn 14.6a

380 thermal\_leaf=epsilon\*sbc\*pow(Tleaf+273,4)\*2;

381 Res = R\_abs - thermal\_leaf - Cp\*HeatConductance\*(Tleaf - Tair) - Lambda\*VaporConductance\*0.5\*(Es(Tleaf)-Ea)/Press; // Residual function: f(Ti), KT Paw (1987)

382 dRes= -4\*epsilon\*sbc\*pow(273+Tleaf,3)\*2-Cp\*HeatConductance\*Tleaf-Lambda\*VaporConductance\*Slope(Tleaf); // derivative of residual: f'(Ti)

383 newTi = Tleaf + Res/dRes; // newton-rhapson iteration

384 iter++;

385 }

386 Tleaf=newTi;

387

388 Transpiration =1000\*VaporConductance\*(Es(Tleaf)-Ea)/Press; //Don't need Lambda - cancels out see eq 14.10 in Campbell and Norman, 1998

389 // umol m-2 s-1. note 1000 converts from moles to umol since units of VaporConductance are moles.

390 }

#### double photomod::CGasExchange::Es (double *Temperature*)[protected]

Saturated vapor pressure at given temperature.

kPa

calculates and returns Saturation vapor pressure (kPa). Campbell and Norman (1998), p 41.

##### Parameters:

|  |  |  |
| --- | --- | --- |
| in | *Temperature* |  |

##### Returns:

saturated vapor pressure (kPa)

Definition at line 455 of file gas\_exchange.cpp.

456 {

463 double result;

464 // a=0.611 kPa, b=17.502 C and c=240.97 C

465 //Units of Es are kPa

466 result=(0.611\*exp(17.502\*Temperature/(240.97+Temperature)));

467 return result;

468 }

#### double photomod::CGasExchange::EvalCi (double *Ci*)[protected]

Calls photosynthesis modules to evaluate Ci dependent calculations returns difference between old Ci and new Ci.

Called by **SearchCi()** to calculate a new value of Ci for the current values of photosynthesis and stomatal conductance determined using parameters from a previous step where the energy balance was solved.

##### See also:

**SearchCi()**

##### Parameters:

|  |  |  |
| --- | --- | --- |
| in | *Ci* | \*\*, estimate of internal CO2 concentration, umol mol-1 |

##### Returns:

the difference between the passed value of Ci (old)and the new one.

Definition at line 561 of file gas\_exchange.cpp.

562 {

573 double newCi;

574

575 if (PlantType.compare("C3")== 0) PhotosynthesisC3(Ci);

576 if (PlantType.compare("C4")== 0) PhotosynthesisC4(Ci);

577 if (abs(StomatalConductance) > eqlTolerance)

578 {

579 newCi = max(1.0,CO2 - AssimilationNet\*(1.6/StomatalConductance+1.37/BoundaryLayerConductance)\*(Press/100.0));

580 }

581 else

582 newCi = max(1.0,CO2 - AssimilationNet\*(1.6/eqlTolerance+1.37/BoundaryLayerConductance)\*(Press/100.0));

583 return (newCi-Ci);

584 }

#### void photomod::CGasExchange::GasEx (void )[protected]

Main module to calculate gas exchange rates.

carries out calculations for photosynthesis and stomatal conductance.

no parameters, returns nothing

##### See also:

**SearchCi()**,

**EnergyBalance()**,

**CalcStomatalConductance()**

##### Returns:

nothing

Definition at line 145 of file gas\_exchange.cpp.

153 {

154 double Tleaf\_old; //previous leaf temperture (for iteration)

155 int iter=1;

156 iter\_total=0;

157 Tleaf = Tair; Tleaf\_old = 0;

158 Ci = 0.7\*CO2;

159 BoundaryLayerConductance = CalcTurbulentVaporConductance();

160 StomatalConductance = CalcStomatalConductance();

161 while ((abs(Tleaf\_old -Tleaf)>0.01) && (iter < maxiter))

162 {

163 Tleaf\_old=Tleaf;

164 Ci=SearchCi(Ci);

165 StomatalConductance=CalcStomatalConductance();

166 EnergyBalance();

167 iter2 =++iter; //iter=iter+1, iter2=iter;

168 if (ConstantLeafTemperature) Tleaf=Tair;

169 }

170

171 }

#### double photomod::CGasExchange::get\_AGross ()[inline]

return gross photosynthesis (umol CO2 m-2 s-1)

Definition at line 166 of file gas\_exchange.h.

#### double photomod::CGasExchange::get\_ANet ()[inline]

return net photosynthesis (umol CO2 m-2 s-1)

Definition at line 165 of file gas\_exchange.h.

#### double photomod::CGasExchange::get\_BoundaryLayerConductance ()[inline]

return boundary layer conductance (mol m-2 s-1)

Definition at line 171 of file gas\_exchange.h.

#### double photomod::CGasExchange::get\_Ci ()[inline]

return internal CO2 concentration (umol mol-1)

Definition at line 169 of file gas\_exchange.h.

#### double photomod::CGasExchange::get\_CiCaRatio ()[inline], [protected]

Ratio between internal and atmospheric CO2.

Definition at line 193 of file gas\_exchange.h.

#### double photomod::CGasExchange::get\_LeafTemperature ()[inline]

return leaf temperature (C)

Definition at line 168 of file gas\_exchange.h.

#### double photomod::CGasExchange::get\_Respiration ()[inline]

return respiration rate (umol CO2 m-2 s-1)

Definition at line 172 of file gas\_exchange.h.

#### double photomod::CGasExchange::get\_StomatalConductance ()[inline]

return stomatal conductance to water vapor (mol m-2 s-1)

Definition at line 170 of file gas\_exchange.h.

#### double photomod::CGasExchange::get\_Transpiration ()[inline]

return transpiration rate (umol H2O m-2 s-1)

Definition at line 167 of file gas\_exchange.h.

#### double photomod::CGasExchange::get\_VPD ()[inline]

return vapor pressure deficit (kpa)

Definition at line 173 of file gas\_exchange.h.

#### double photomod::CGasExchange::Min (double *a*, double *b*, double *c*)[inline], [protected]

Finds minimum of three numbers.

Definition at line 228 of file gas\_exchange.h.

#### double photomod::CGasExchange::minh (double *fn1*, double *fn2*, double *theta2*)[protected]

Hyperbolic minimum.

##### Parameters:

|  |  |  |
| --- | --- | --- |
| in | *fn1* | first value to be compared for min |
| in | *fn2* | second value to be compared for min |
| in | *theta2* | curvature factor |

##### Returns:

hyperbolic minimum

Definition at line 617 of file gas\_exchange.cpp.

618 {

619

627 double x, res;

628

629 x = ((fn1+fn2)\*(fn1+fn2)-4\*theta2\*fn1\*fn2);

630 if (x<0)

631 {

632 res = min(fn1,fn2);

633 return res;

634 }

635 if (theta2==0.0)

636 {

637 res= fn1\*fn2/(fn1+fn2);

638 return res;

639 }

640 else

641 {

642 res = ((fn1+ fn2) - sqrt(x))/(2\*theta2); // hyperbolic minimum

643 return res;

644 }

645 }

#### void photomod::CGasExchange::PhotosynthesisC3 (double *Ci*)[protected]

Function to calculate C3 photosynthesis.

Calculates photosynthesis for C3 plants

Uses as input incident PhotoFluxDensity, Air temp in C, CO2 in ppm, RH in per percent.

##### See also:

**SetVal()**

##### Parameters:

|  |  |  |
| --- | --- | --- |
| in | *Ci* | - internal CO2 concentration, umol mol-1 |

##### Returns:

nothing.

< **Curvature** -factor of Av and Aj colimitation

< **Kc25** , MM Constant of rubisco for CO2 of C3 plants (de Pury and Farquar, 1997) (umol m-2 s-1)

< **Ko25** , MM Constant of rubiscuo for O2 from above reference (umol m-2 s-1)

< **Eac** , Energy Activation kJ mol-1

< **Eao** , activation energy values

Definition at line 172 of file gas\_exchange.cpp.

183 {

184 //parameters for C3 Photosythesis;

185 const double curvature=0.999 ;

186

187 const int Kc25 = 404;

188 const int Ko25 = 278;

189 const long Eac = 59400;

190

191 const long Eao = 36000;

192 //\*\* \endcode

193 // These variables hold temporary calculations

194 double alpha, Kc, Ko, gamma, Ia,Jmax, Vcmax, TPU, J, Av, Aj, Ap, Ac, Km, Ca, Cc, P;

195 gamma = 36.9 + 1.88\*(Tleaf-25)+0.036\*Square(Tleaf-25); // CO2 compensation point in the absence of mitochondirial respiration, in ubar}

196 //\* Light response function parameters \*/

197 Ia = PhotoFluxDensity\*(1-scatt); //\* absorbed irradiance \*/

198 alpha = (1-f)/2; // \*!apparent quantum efficiency, params adjusted to get value 0.3 for average C3 leaf

199

200 AssimilationNet = 0;

201

202 //\* other input parameters and constants \*/

203 P = Press/100; //Press is kPa. Used to convert mole fraction to partial pressure

204 Ca = CO2\*P; //\* conversion to partial pressure \*/

205 Kc = Kc25\*exp(Eac\*(Tleaf-25)/(298\*R\*(Tleaf+273)));

206 Ko = Ko25\*exp(Eao\*(Tleaf-25)/(298\*R\*(Tleaf+273)));

207 Km = Kc\*(1+O/Ko); //\* effective M-M constant for Kc in the presence of O2 \*/

208 DarkRespiration = sParms.Rd25\*exp(sParms.Ear\*(Tleaf-25)/(298\*R\*(Tleaf+273)));

209 Jmax = sParms.Jm25\*exp(((Tleaf-25)\*sParms.Eaj)/(R\*(Tleaf+273)\*298))\*

210 (1+exp((sParms.Sj\*298-sParms.Hj)/(R\*298)))/

211 (1+exp((sParms.Sj\*(Tleaf+273)-sParms.Hj)/(R\*(Tleaf+273)))); // de Pury 1997

212 Vcmax = sParms.Vcm25\*exp(((Tleaf-25)\*sParms.EaVc)/(R\*(Tleaf+273)\*298))\*

213 (1+exp((sParms.Sv\*298-sParms.Hv)/(R\*298)))/

214 (1+exp((sParms.Sv\*(Tleaf+273)-sParms.Hv)/(R\*(Tleaf+273)))); // Used peaked response, DHF

215 TPU = sParms.TPU25\*exp(sParms.Eap\*(Tleaf-25)/(298\*R\*(Tleaf+273)));

216 Cc = Ci; // assume infinite gi

217

218 StomatalConductance = CalcStomatalConductance(); // Initial value

219 BoundaryLayerConductance= CalcTurbulentVaporConductance();

220 Av = (Vcmax\*(Cc-gamma))/(Cc+Km);

221 J = (((alpha\*Ia + Jmax) - sqrt(Square(alpha\*Ia+Jmax) - 4\*alpha\*Ia\*(Jmax)\*sParms.Theta)) / (2\*sParms.Theta)) ;

222 Aj = J\*(Cc-gamma)/(4\*(Cc+2\*gamma));

223 Ap = 3\*TPU;

224 Ac = ((Av+Aj) - sqrt(Square(Av+Aj)-4\*curvature\*Av\*Aj))/(2\*curvature); // curvatureaccount for colimitation between Av and Aj \*/

225 if (Cc > gamma)

226 AssimilationNet = min(Ac, Ap) -DarkRespiration;

227 else

228 {

229 AssimilationNet = Av-DarkRespiration;

230 }

231

232 AssimilationGross = max(AssimilationNet+DarkRespiration,0.0);

233 StomatalConductance = CalcStomatalConductance(); // Update StomatalConductance using new value of AssimilationNet

234

235 }

#### void photomod::CGasExchange::PhotosynthesisC4 (double *Ci*)[protected]

Function to calculate C4 photosynthesis.

Calculates photosynthesis for C4 plants.

Requires Incident PhotoFluxDensity, Air temp in C, CO2 in ppm, RH in percent

##### See also:

**SetVal()**

##### Parameters:

|  |  |  |
| --- | --- | --- |
| in | *Ci* | - internal CO2 concentration, umol mol-1 |

##### Returns:

nothing

< **curvature** factor of Av and Aj colimitation

< **Kc25** , Michaelis constant of rubisco for CO2 of C4 plants (2.5 times that of tobacco), ubar, Von Caemmerer 2000

< **Ko25** , Michaelis constant of rubisco for O2 (2.5 times C3), mbar

<**Kp** , **Kc** , **Ko** , **Km** , Calculated Michaelis params as a function of temperature

Definition at line 237 of file gas\_exchange.cpp.

247 {

248 const double curvature=0.995;

249

250

251 const int Kc25 = 650,

252 Ko25 = 450,

253 Kp25 = 57; //\*!< \b Kp25, Michaelis constant for PEP caboxylase for CO2 - was 60 in Kim's paper \*/

254 const long Eao = 36000; //\*!< \b EAO, activation energy for Ko \*/

255 const int Vpr25 = 80; //\*!< \b Vpr25, PEP regeneration limited Vp at 25C, value adopted from vC book \*/

256 const double gbs = 0.003; //\*!< \b gbs, bundle sheath conductance to CO2, umol m-2 s-1 gbs x Cm is the inward diffusion of CO2 into the bundle sheath \*/

257 const double x = 0.4; //\*!< \b x Partitioning factor of J, yield maximal J at this value \*/

258 const double alpha = 0.001; //\*!< \b alpha, fraction of PSII activity in the bundle sheath cell, very low for NADP-ME types \*/

259 const double gi = 5.0; //\*!< \b gi, conductance to CO2 from intercelluar to mesophyle, mol m-2 s-1, assumed was 1, changed to 5 as per Soo 6/2012\*/

260 const double beta = 0.99; //\*!< \b beta, smoothing factor \*/

261 const double gamma1 = 0.193; //\*!< \b gamma1, half the reciprocal of rubisco specificity, to account for O2 dependence of CO2 comp point, note that this become the same as that in C3 model when multiplied by [O2] \*/

262

263 double Kp, Kc, Ko, Km;

264 double Ia, I2; // secondary calculated light variables

265 double Vpmax, Jmax, Vcmax, Eac, Om, Rm, J, Ac1, Ac2, Ac, Aj1,

266 Aj2, Aj, Vp1, Vp2, Vp, P, Ca, Cm, Vpr,

267 Os, GammaStar, Gamma, a1, b1, c1; //secondary calculated variables

268

269 //\* Light response function parameters \*/

270 Ia = PhotoFluxDensity\*(1-scatt); //\* absorbed irradiance \*/

271 I2 = Ia\*(1-f)/2; //\* useful light absorbed by PSII \*/

272 //\* other input parameters and constants \*/

273 P = Press/100;

274 Ca = CO2\*P; //\* conversion to partial pressure Atmospheric partial pressure of CO2, kPa\*/

275 Om = O; //\* mesophyle O2 partial pressure \*/

276 Eac=sParms.EaVc;

277

278 Kp = Kp25\*pow(Q10,(Tleaf-25.0)/10.0);

279 Vpr = Vpr25\*pow(Q10,(Tleaf-25.0)/10.0);

280 Kc = Kc25\*exp(Eac\*(Tleaf-25)/(298\*R\*(Tleaf+273))); //Kc adjusted for temperature

281 Ko = Ko25\*exp(Eao\*(Tleaf-25)/(298\*R\*(Tleaf+273)));

282 Km = Kc\*(1+Om/Ko); //\* effective M-M constant for Kc in the presence of O2 \*/

283 DarkRespiration = sParms.Rd25\*exp(sParms.Ear\*(Tleaf-25)/(298\*R\*(Tleaf+273)));

284 // The following are Arrhenius Equations for parameter temperature dependencies

285 // Vpm25 (PEPC activity rate) , Vcm25 (Rubisco Capacity rate) and Jm25 (Whole chain electron transport rate) are the rates at 25C for Vp, Vc and Jm

286 Vpmax = sParms.Vpm25\*exp(sParms.EaVp\*(Tleaf-25)/(298\*R\*(Tleaf+273)));

287 Vcmax = sParms.Vcm25\*exp(sParms.EaVc\*(Tleaf-25)/(298\*R\*(Tleaf+273)));

288 Jmax = sParms.Jm25\*exp((((Tleaf+273)-298)\*sParms.Eaj)/(R\*(Tleaf+273)\*298))\*(1+exp((sParms.Sj\*298-sParms.Hj)/(R\*298)))

289 /(1+exp((sParms.Sj\*(Tleaf+273)-sParms.Hj)/(R\*(Tleaf+273.0))));

290 Rm = 0.5\*DarkRespiration;

291

292 Cm=Ci; //\* mesophyle CO2 partial pressure, ubar, one may use the same value as Ci assuming infinite mesohpyle conductance \*/

293 double gs\_last=0;

294

295 StomatalConductance = CalcStomatalConductance();

296 Vp1 = (Cm\*Vpmax)/(Cm+Kp); //\* PEP carboxylation rate, that is the rate of C4 acid generation Eq 1 in Kim 2007\*/

297 Vp2 = Vpr;

298 Vp = \_\_max(\_\_min(Vp1, Vp2),0);

299 //\* Enzyme limited A (Rubisco or PEP carboxylation \*/

300 Ac1 = (Vp+gbs\*Cm-Rm);

301 Ac2 = (Vcmax-DarkRespiration);

302 //\* Quadratic expression to solve for Ac \*/

303 a1 = 1-(alpha/0.047)\*(Kc/Ko);

304 b1 = -(Ac1 + Ac2 + gbs\*Km + (alpha/0.047)\*(gamma1\*Vcmax + DarkRespiration\*Kc/Ko));

305 c1 = Ac1\*Ac2-(Vcmax\*gbs\*gamma1\*Om+DarkRespiration\*gbs\*Km);

306 Ac = QuadSolnLower(a1,b1,c1);

307 Ac = \_\_min(Ac1,Ac2);

308 //\* Light and electron transport limited A mediated by J \*/

309 J=minh(I2,Jmax,sParms.Theta); //\* rate of electron transport \*/

310 Aj1 = (x\*J/2-Rm+gbs\*Cm); // Eq 4 in Kim, 2007

311 Aj2 = (1-x)\*J/3-DarkRespiration; //Eq 4 in Kim, 2007

312 Aj = \_\_min(Aj1,Aj2); //Eq 4 in Kim, 2007

313 AssimilationNet = ((Ac+Aj) - sqrt(Square(Ac+Aj)-4\*beta\*Ac\*Aj))/(2\*beta); //\* smooting the transition between Ac and Aj \*/

314 AssimilationNet=minh(Ac,Aj, curvature);

315 gs\_last=StomatalConductance;

316 Os = alpha\*AssimilationNet/(0.047\*gbs)+Om; //\* Bundle sheath O2 partial pressure, mbar \*/

317 GammaStar = gamma1\*Os;

318 Gamma = (DarkRespiration\*Km + Vcmax\*GammaStar)/(Vcmax-DarkRespiration);

319 AssimilationGross = \_\_max(0, AssimilationNet + DarkRespiration);

320

321 }

#### double photomod::CGasExchange::QuadSolnLower (double *a*, double *b*, double *c*)[protected]

Lower part of quadratic equation solution.

solves the lower part of the quadratic equation ax2+bx=c

##### Parameters:

|  |  |  |
| --- | --- | --- |
| in | *a* |  |
| in | *b* |  |
| in | *c* |  |

##### Returns:

lower portion of x

Definition at line 602 of file gas\_exchange.cpp.

603 {

611 if (a==0) return 0;

612 else if ((b\*b - 4\*a\*c) < 0) return -b/a; //imaginary roots

613 else return (-b-sqrt(b\*b-4\*a\*c))/(2\*a);

614 }

#### double photomod::CGasExchange::QuadSolnUpper (double *a*, double *b*, double *c*)[protected]

Upper part of quadratic equation solution.

solves the uppper part of the quadratic equation ax2+bx2=c

##### Parameters:

|  |  |  |
| --- | --- | --- |
| in | *a* |  |
| in | *b* |  |
| in | *c* |  |

##### Returns:

lower portion of x

Definition at line 587 of file gas\_exchange.cpp.

588 {

589

597 if (a==0) return 0;

598 else if ((b\*b - 4\*a\*c) < 0) return -b/a; //imaginary roots

599 else return (-b+sqrt(b\*b-4\*a\*c))/(2\*a);

600 }

#### double photomod::CGasExchange::SearchCi (double *CO2i*)[protected]

called iterively to find optimal internal CO2 concentration returns optimal internal CO2 concentration (CO2i)

does a secant search to find the optimal internal CO2 concentration (ci) Calls:

##### See also:

**EvalCi()**

##### Parameters:

|  |  |  |
| --- | --- | --- |
| in | *CO2i* | - internal CO2 concentration, (umol mol-1) |

##### Returns:

Ci (umol mol-1)

Definition at line 488 of file gas\_exchange.cpp.

489 {

498 int iter;

499 double fprime, Ci1, Ci2, Ci\_low, Ci\_hi, Ci\_m;

500 double temp;

501 Ci1 = CO2i;

502 Ci2 = CO2i + 1.0;

503 Ci\_m = (Ci1+Ci2)/2.0;

504 iter\_Ci = 0;

505 iter = 0;

506 isCiConverged = true;

507

508 do

509 {

510 iter++;

511 //Secant search method

512 if (abs(Ci1-Ci2) <= errTolerance) {break;}

513 if (iter >= maxiter)

514 {

515 isCiConverged = false;

516 break;

517 }

518 fprime = (EvalCi(Ci2)-EvalCi(Ci1))/(Ci2-Ci1); // f'(Ci)

519 if (fprime != 0.0)

520 {

521 Ci\_m = max(errTolerance, Ci1-EvalCi(Ci1)/fprime);

522 }

523 else

524 Ci\_m = Ci1;

525 Ci1 = Ci2;

526 Ci2 = Ci\_m;

527 temp=EvalCi(Ci\_m);

528 double temp2=maxiter;

529 } while ((abs(EvalCi(Ci\_m)) >= errTolerance) || (iter < maxiter));

530

531

532

533

534 // C4 photosynthesis fails to converge at low soil water potentials using secant search, 6/8/05 SK

535 // Bisectional type search is slower but more secure

536 //Bisectional search

537 if (iter > maxiter)

538 {

539 Ci\_low = 0.0;

540 Ci\_hi = 2.0\*CO2;

541 isCiConverged = false;

542

543 while (abs(Ci\_hi-Ci\_low) <= errTolerance || iter > (maxiter\*2))

544 {

545 Ci\_m = (Ci\_low + Ci\_hi)/2;

546 if (abs(EvalCi(Ci\_low)\*EvalCi(Ci\_m)) <= eqlTolerance) break;

547 else if (EvalCi(Ci\_low)\*EvalCi(Ci\_m) < 0.0) {Ci\_hi = max(Ci\_m, errTolerance);}

548 else if (EvalCi(Ci\_m)\*EvalCi(Ci\_hi) < 0.0) {Ci\_low = max(Ci\_m, errTolerance);}

549 else {isCiConverged = false; break;}

550 }

551

552 }

553

554 CO2i = Ci\_m;

555 Ci\_Ca = CO2i/CO2;

556 iter\_Ci = iter\_Ci + iter;

557 iter\_total = iter\_total + iter;

558 return CO2i;

559

560 }

#### void photomod::CGasExchange::SetParams (tParms \* *sParms*)

used to pass structure with parameters from the interface to the calculator

##### See also:

**SetParams** for definitions

Definition at line 59 of file gas\_exchange.cpp.

60 {

61

62

64 PlantType=sParms->Type;

65 if (PlantType.compare("C4")==0)

66 {

67

68 this->sParms.Vcm25 = sParms->Vcm25;

69 this->sParms.Jm25 = sParms->Jm25;

70 this->sParms.Vpm25 = sParms->Vpm25;

71 this->sParms.Rd25 = sParms->Rd25;

72 this->sParms.Theta = sParms->Theta;

73 this->sParms.EaVc = sParms->EaVc;

74 this->sParms.Eaj = sParms->Eaj;

75 this->sParms.Hj = sParms->Hj;

76 this->sParms.Sj = sParms->Sj;

77 this->sParms.EaVp = sParms->EaVp;

78 this->sParms.Ear = sParms->Ear;

79 this->sParms.g0 = sParms->g0;

80 this->sParms.g1 = sParms->g1;

81 this->sParms.stomaRatio = sParms->stomaRatio;

82 this->sParms.LfWidth = sParms->LfWidth;

83

84 }

85

86 if (PlantType.compare("C3")==0)

87 {

88

89 this->sParms.Vcm25 = sParms->Vcm25;

90 this->sParms.Jm25 = sParms->Jm25;

91 this->sParms.TPU25 = sParms->TPU25;

92 this->sParms.Rd25 = sParms->Rd25;

93 this->sParms.Theta = sParms->Theta;

94 this->sParms.EaVc = sParms->EaVc;

95 this->sParms.Eaj = sParms->Eaj;

96 this->sParms.Hj = sParms->Hj;

97 this->sParms.Sj = sParms->Sj;

98 this->sParms.Hv = sParms->Hv;

99 this->sParms.Sv = sParms->Sv;

100 this->sParms.Eap = sParms->Eap;

101 this->sParms.Ear = sParms->Ear;

102 this->sParms.g0 = sParms->g0;

103 this->sParms.g1 = sParms->g1;

104 this->sParms.stomaRatio = sParms->stomaRatio;

105 this->sParms.LfWidth = sParms->LfWidth;

106

107

108 }

109

110

111 }

#### void photomod::CGasExchange::SetVal (double *PhotoFluxDensity*, double *Tair*, double *CO2*, double *RH*, double *wind*, double *Press*, bool *ConstantTemperature*)

sets input values for calculations for a particular set of environmental variables

Sets environment variables for a single execution of the module.

Calls **GasEx()** to calculate photosynthetic rate and stomatal conductance.

##### Parameters:

|  |  |  |
| --- | --- | --- |
| in | *PhotoFluxDensity* | Photosynthetic Flux Density (umol Quanta m-2 s-1) (check) |
| in | *Tair* | Air Temperature (C) |
| in | *CO2* | CO2 concentration of the air (umol mol-1) |
| in | *RH* | Relative Humidity (%) |
| in | *wind* | Windspeed at 2.5 m, m s-1 |
| in | *Press* | Atmospheric pressure (kpa m-2) |
| in | *ConstantTemperature* | boolian if true, leaf temperature=air temperature when calculating gas exchange |

##### Returns:

nothing

Definition at line 113 of file gas\_exchange.cpp.

129 {

130 this->PhotoFluxDensity = PhotoFluxDensity;

131 double PAR = (PhotoFluxDensity/4.55); //PAR is watts m-2

132 double NIR = PAR; // If total solar radiation unavailable, assume NIR the same energy as PAR waveband

133 this->R\_abs = (1-scatt)\*PAR + 0.15\*NIR + 2\*(epsilon\*sbc\*pow(Tair+273,4)); // times 2 for projected area basis

134 // shortwave radiation (PAR (=0.85) + NIR (=0.15) solar radiation absorptivity of leaves: =~ 0.5

135 //transfer variables to local scope

136 this->CO2 = CO2;

137 this->RH = \_\_min(100.0, \_\_max(RH, 10.0))/100;

138 this->Tair = Tair;

139 this->wind = wind;

140 this->Press = Press;

141 ConstantLeafTemperature=ConstantTemperature;

142 GasEx(); // Gas exchange calculations here

143 }

#### double photomod::CGasExchange::Slope (double *Temperature*)[protected]

Slope of the vapor pressure curve.

Calculates the slope of the sat vapor pressure curve: first order derivative of Es with respect to T.

##### Parameters:

|  |  |  |
| --- | --- | --- |
| in | *Temperature* | (C) |

##### See also:

**Es()**

##### Returns:

slope of the vapor pressure curve kPa T-1

Definition at line 470 of file gas\_exchange.cpp.

480 {

481 double VPSlope;

482 // units of b and c are degrees C

483 const double b= 17.502; const double c= 240.97;

484 VPSlope=(Es(Temperature)\*(b\*c)/Square(c+Temperature)/Press);

485 return VPSlope;

486 }

#### double photomod::CGasExchange::Square (double *a*)[inline], [protected]

Squares number.

Definition at line 227 of file gas\_exchange.h.

### Member Data Documentation

#### double photomod::CGasExchange::AssimilationGross[protected]

Gross photosynthesis (umol CO2 m-2 s-1) (Adjusted for respiration)

Definition at line 205 of file gas\_exchange.h.

#### double photomod::CGasExchange::AssimilationNet[protected]

Net photosynthesis (umol CO2 m-2 s-1)

Definition at line 205 of file gas\_exchange.h.

#### double photomod::CGasExchange::BoundaryLayerConductance[protected]

Boundary layer conductance umol m-2 s-1.

Definition at line 205 of file gas\_exchange.h.

#### double photomod::CGasExchange::Ci[protected]

Internal CO2 concentration umol mol-1.

Definition at line 205 of file gas\_exchange.h.

#### double photomod::CGasExchange::Ci\_Ca[protected]

Ratio of internal to external CO2, unitless.

Definition at line 205 of file gas\_exchange.h.

#### double photomod::CGasExchange::CO2[protected]

CO2 concentration (umol mol-1 air)

Definition at line 195 of file gas\_exchange.h.

#### bool photomod::CGasExchange::ConstantLeafTemperature[protected]

if true, uses constant temperature - if true, does not solve for leaf temperature \*/

Definition at line 223 of file gas\_exchange.h.

#### double photomod::CGasExchange::DarkRespiration[protected]

Plant respiration umol m-2 s-1.

Definition at line 205 of file gas\_exchange.h.

#### double photomod::CGasExchange::eqlTolerance[protected]

Equality tolerance.

Definition at line 217 of file gas\_exchange.h.

#### double photomod::CGasExchange::errTolerance[protected]

Error tolerance for iterations.

Definition at line 216 of file gas\_exchange.h.

#### bool photomod::CGasExchange::isCiConverged[protected]

True if Ci (internal CO2 concentration) iterations have converged.

Definition at line 226 of file gas\_exchange.h.

#### int photomod::CGasExchange::iter1[protected]

Definition at line 224 of file gas\_exchange.h.

#### int photomod::CGasExchange::iter2[protected]

Iteration counters.

Definition at line 224 of file gas\_exchange.h.

#### int photomod::CGasExchange::iter\_Ci[protected]

Iteration value for Ci umol mol-1, internal CO2 concentration.

Definition at line 225 of file gas\_exchange.h.

#### int photomod::CGasExchange::iter\_total[protected]

Total number of iterations.

Definition at line 219 of file gas\_exchange.h.

#### double photomod::CGasExchange::PhotoFluxDensity[protected]

Photosynthetic Flux Density (umol photons m-2 s-1.

Definition at line 195 of file gas\_exchange.h.

#### \* std::string photomod::CGasExchange::PlantType[protected]

For C3 or C4 designation \*/.

##### Parameters:

|  |  |
| --- | --- |
| *PlantType* | string that holds the type of plant, C3 or C4 |

Definition at line 222 of file gas\_exchange.h.

#### double photomod::CGasExchange::Press[protected]

Air pressure (kPa)

Definition at line 195 of file gas\_exchange.h.

#### double photomod::CGasExchange::R\_abs[protected]

Absorbed incident radiation (watts m-2)

Definition at line 195 of file gas\_exchange.h.

#### double photomod::CGasExchange::RH[protected]

Relative Humidity (%, i.e., 80)

Definition at line 195 of file gas\_exchange.h.

#### struct photomod::CGasExchange::tParms photomod::CGasExchange::sParms

Variable of type **tParms**.

#### double photomod::CGasExchange::StomatalConductance[protected]

Stomatal conductance umol m-2 s-1.

Definition at line 205 of file gas\_exchange.h.

#### double photomod::CGasExchange::Tair[protected]

Air temperature at 2m, (C)

Definition at line 195 of file gas\_exchange.h.

#### double photomod::CGasExchange::Tleaf[protected]

Leaf temperature C.

Definition at line 205 of file gas\_exchange.h.

#### double photomod::CGasExchange::Transpiration[protected]

Transpiration mol H2O m-2 s-1.

Definition at line 205 of file gas\_exchange.h.

#### double photomod::CGasExchange::VPD[protected]

Vapor Pressure Density, kPa \*/.

Definition at line 205 of file gas\_exchange.h.

#### double photomod::CGasExchange::width[protected]

Leaf width (m)

Definition at line 195 of file gas\_exchange.h.

#### double photomod::CGasExchange::wind[protected]

Windspeed at 2 meters (km s-1)

Definition at line 195 of file gas\_exchange.h.

#### The documentation for this class was generated from the following files:

* **gas\_exchange.h**
* **gas\_exchange.cpp**

## tparms Struct Reference

Structure to hold plant parameters for the photosynthesis model.

#include <gas\_exchange.h>

### Detailed Description

Structure to hold plant parameters for the photosynthesis model.

Some parameters are specific for C3 or C4 type Plants

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

* **gas\_exchange.h**

## photomod::CGasExchange::tParms Struct Reference

parameter description

#include <gas\_exchange.h>

### Public Attributes

* string **ID**
* std::string **species**
* std::string **Type**
* double **Vcm25**
* double **Jm25**
* double **Vpm25**
* double **TPU25**
* double **Rd25**
* double **Theta**
* double **EaVc**
* double **Eaj**
* double **Hj**
* double **Sj**
* double **Hv**
* double **EaVp**
* double **Sv**
* double **Eap**
* double **Ear**
* double **g0**
* double **g1**
* double **stomaRatio**
* double **LfWidth**
* double **LfAngFact**
* std::string **Remark**

### Detailed Description

parameter description

##### Parameters:

|  |  |
| --- | --- |
| *ID* | 1 Name of plant (string) |
| *species* | 2 Species Name (string) |
| *type* | 3 'C3' or 'C4' (string) |
| *Vcm25* | 4 Photosynthetic Rubisco Capacity at 25C (umol m-2 s-1) |
| *Jm25* | 5 Potential Rate of electron transport at 25C (umol m-2 s-1) |
| *Vpm25* | 6 C4 Carboxylation rate at 25C (C4, umol m-2 s-1) |
| *TPU25* | 7 Rate if Triose Phosphate Utilization at 25C (C3, umol m-2 s-1) |
| *RD25* | 8 Mitochondrial respiration in the light at 25C (umol m-2 s-1) |
| *Theta* | 9 Initial slope of CO2 response (umol m2 s-1) - de Pury (1997) |
| *EaVc* | 10 Activation energy for Arrhenius function used to calculate temperature dependence for Vcmax (kJ mol-1) |
| *Eaj* | 11 Activation energy for Arrhenius function used to calculate temperature dependence for J (kJ mol-1) |
| *Hj* | 12 Curvature parameter of the temperature dpendence of Jmax (kJ mol-1) |
| *Sj* | 13 Electron transport temperature response parameter for Jmax (J mole-1 K-1) |
| *Hv* | 14 Curvature parameter of the temperature dependence of Vcmax (J mole-1) |
| *EaVp* | 15 Activation energy for Arrhenius function used to calculate temperature dependence for Vpmax (kJ mol-1) |
| *Sv* | 16 Electron transport temperature response parameter for Vcmax (J mole-1 K-1) |
| *EAP* | 17 Activation energy for Arrhenius function used to calculate temperature dependence for TPU (kJ mol-1) |
| *EAR* | 18 Activation energy for Arrhenius function used to calculate temperature dependence for respiration (kJ mol-1) |
| *g0* | 19 Minimum stomatal conductance to water vapor at the light compensation point in the BWB model (mol m-2 s-1) |
| *g1* | 20 Empirical coefficient for the sensitivity of StomatalConductance to A, Cs and hs in BWB model (no units) |
| *StomRatio* | 21 Stomatal Ratio (fraction) |
| *LfWidth* | 22 Leaf Width (m) |
| *LfAngFact* | 23 Leaf Angle Factor |
| *Remark* | 24 Text |

Definition at line 132 of file gas\_exchange.h.

### Member Data Documentation

#### double photomod::CGasExchange::tParms::Eaj

Definition at line 139 of file gas\_exchange.h.

#### double photomod::CGasExchange::tParms::Eap

Definition at line 139 of file gas\_exchange.h.

#### double photomod::CGasExchange::tParms::Ear

Definition at line 139 of file gas\_exchange.h.

#### double photomod::CGasExchange::tParms::EaVc

Definition at line 139 of file gas\_exchange.h.

#### double photomod::CGasExchange::tParms::EaVp

Definition at line 139 of file gas\_exchange.h.

#### double photomod::CGasExchange::tParms::g0

Definition at line 139 of file gas\_exchange.h.

#### double photomod::CGasExchange::tParms::g1

Definition at line 139 of file gas\_exchange.h.

#### double photomod::CGasExchange::tParms::Hj

Definition at line 139 of file gas\_exchange.h.

#### double photomod::CGasExchange::tParms::Hv

Definition at line 139 of file gas\_exchange.h.

#### string photomod::CGasExchange::tParms::ID

Definition at line 134 of file gas\_exchange.h.

#### double photomod::CGasExchange::tParms::Jm25

Definition at line 139 of file gas\_exchange.h.

#### double photomod::CGasExchange::tParms::LfAngFact

Definition at line 139 of file gas\_exchange.h.

#### double photomod::CGasExchange::tParms::LfWidth

Definition at line 139 of file gas\_exchange.h.

#### double photomod::CGasExchange::tParms::Rd25

Definition at line 139 of file gas\_exchange.h.

#### std::string photomod::CGasExchange::tParms::Remark

Definition at line 159 of file gas\_exchange.h.

#### double photomod::CGasExchange::tParms::Sj

Definition at line 139 of file gas\_exchange.h.

#### std::string photomod::CGasExchange::tParms::species

Definition at line 135 of file gas\_exchange.h.

#### double photomod::CGasExchange::tParms::stomaRatio

Definition at line 139 of file gas\_exchange.h.

#### double photomod::CGasExchange::tParms::Sv

Definition at line 139 of file gas\_exchange.h.

#### double photomod::CGasExchange::tParms::Theta

Definition at line 139 of file gas\_exchange.h.

#### double photomod::CGasExchange::tParms::TPU25

Definition at line 139 of file gas\_exchange.h.

#### std::string photomod::CGasExchange::tParms::Type

Definition at line 136 of file gas\_exchange.h.

#### double photomod::CGasExchange::tParms::Vcm25

Definition at line 139 of file gas\_exchange.h.

#### double photomod::CGasExchange::tParms::Vpm25

Definition at line 139 of file gas\_exchange.h.

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

* **gas\_exchange.h**

# File Documentation

## gas\_exchange.cpp File Reference

Contains code to simulate gas exchange.

#include "stdafx.h"

#include "gas\_exchange.h"

#include <cmath>

### Namespaces

* **photomod**

### Macros

* #define **R**  8.314

***R*** *idealgasconstant*

* #define **maxiter**  200

***maxiter*** *maximum number of iterations*

* #define **epsilon**  0.97

*epsilon emissivity See Campbell and Norman, 1998, page 163 (CHECK)*

* #define **sbc**  5.6697e-8

*stefan-Boltzmann constant Wm-2 k-4.*

* #define **scatt**  0.15

*leaf reflectance + transmittance*

* #define **f**  0.15

*spectral correction*

* #define **O**  205.0

*Oxygen partial pressure gas units are mbar.*

* #define **Q10**  2.0

*Q10 factor.*

### Detailed Description

Contains code to simulate gas exchange.

### Macro Definition Documentation

#### #define epsilon  0.97

epsilon emissivity See Campbell and Norman, 1998, page 163 (CHECK)

Definition at line 21 of file gas\_exchange.cpp.

#### #define f  0.15

spectral correction

Definition at line 24 of file gas\_exchange.cpp.

#### #define maxiter  200

**maxiter** maximum number of iterations

Definition at line 20 of file gas\_exchange.cpp.

#### #define O  205.0

Oxygen partial pressure gas units are mbar.

Definition at line 25 of file gas\_exchange.cpp.

#### #define Q10  2.0

Q10 factor.

Definition at line 26 of file gas\_exchange.cpp.

#### #define R  8.314

**R** idealgasconstant

Definition at line 19 of file gas\_exchange.cpp.

#### #define sbc  5.6697e-8

stefan-Boltzmann constant Wm-2 k-4.

Actually varies somewhat with temperature

Definition at line 22 of file gas\_exchange.cpp.

#### #define scatt  0.15

leaf reflectance + transmittance

Definition at line 23 of file gas\_exchange.cpp.

## gas\_exchange.h File Reference

function headers for calculations

#include "stdafx.h"

### Classes

* class **photomod::CGasExchange**

*Class for gas exchange calculations*

* *This class simulates gas exchange in plant leaves for C3 and C4 plants.* struct **photomod::CGasExchange::tParms**

### *parameter description* Namespaces

* **photomod**

### Detailed Description

function headers for calculations

## Photosynthesis(notManaged).cpp File Reference

Defines the entry point for the console application.

#include "stdafx.h"

#include "gas\_exchange.h"

#include <iostream>

#include <fstream>

#include <algorithm>

#include <sstream>

### Namespaces

* **photomod**

### Functions

* int **\_tmain** (int argc, \_TCHAR \*argv[])

### Detailed Description

Defines the entry point for the console application.

##### Author:

$Author

### Function Documentation

#### int \_tmain (int *argc*, \_TCHAR \* *argv*[])

these are the variables for the interface

< **DataLine** , holds line of data read from file with climate data

< **Remark** , remark from parameter file

< **context** , pointer needed to use strtok\_s function

< **ConstantTemperature** , if set to 1, model does not solve for leaf temperature

< **thisParms** , object to hold parameters sent to photosynthesis module

< **pDelim** , -pointer to character (delimiter) that separates entries in the parameter file

< **pnt** , -pointer to the next word to be read from parameter file

< **CharTest** -variable to test if there are characters in the line of data (indicates end of data)

< **found** -Boolean to indicate the species line was found in the parameter file

< **temp** -temporary variable for holding string objects

* these variables hold data sent to model

< **PFD** light umol ppfd m-2 s-1-

< **Temperature** leaf temperature C

< **ParamFile** , **DataFile** - files to hold parameters and variables for a single run

< **OutputFile** holds data output from photosynthesis module

< **MyLeafGasEx** - define a gas exchange object type

< **Species** of plant for calculatioins

Definition at line 63 of file Photosynthesis(notManaged).cpp.

64 {

67 string DataLine;

68 string Remark;

69 char\* context = NULL;

70 bool ConstantTemperature;

71 photomod::CGasExchange::tParms thisParms;

72 const char \*pDelim=",";

73 char \* pnt;

74 char CharTest ='A';

75 bool found=false;

76 string temp ;

77

78 double PFD,

79 Temperature,

80 RelativeHumidity,Wind, CO2, Pressure=100;

81

82 ifstream ParamFile, DataFile;

83 ofstream OutputFile;

84 //Define a pointer to a new variable as a GasExchange Object

85 photomod::CGasExchange \*MyLeafGasEx;

86 // Create a new GasExchange Object

87 MyLeafGasEx= new CGasExchange(); //create a new gas exchange object on the stack

88

89 //variables to hold results from gas exchange module.

90 double Anet, VPD,Agross,LeafTemperature, Respiration, InternalCO2, StomatalConductance,

91 BoundaryLayerConductance, Transpiration;

92

93

94 //assign defaults in case user does not want to enter other information (DataLine is empty)

95 string DataFileName="ClimateIn.dat", OutputFileName="Results.dat";

96 string Species ="Maize";

97

98 // Get datafile name, and species name if entered by user.

99 cout << "enter name of file with input data and species name separated by commas:" <<endl

100 << "hit Enter with empty string for defaults" <<endl;

101 getline(cin,DataLine);

102 if (!DataLine.empty()) //if empty defaults to string names assigned above

103 {

104

105 pnt=strtok\_s((char\*)DataLine.c\_str(), pDelim, &context ); //pnt is a pointer to the last recently found token in the string

106 DataFileName.assign(pnt); // first token is a file name

107 pnt=strtok\_s(NULL, pDelim, &context ); //get ready for the next token by getting pointer to recently found token in the string

108 Species.assign(pnt); // next token is species

109 Species.erase(remove(Species.begin(),Species.end(),' '),Species.end()); //remove blanks from species name in case any are present

110 pnt=NULL; //finished with these two

111 }

112 // open file with parameters and data

113 ParamFile.open("parameters.csv", std::ifstream::in);

114

115 if (!ParamFile)

116 {

117 std::cerr << "Parameter file not found \n";

118 return 0;

119 }

120

121 DataFile.open(DataFileName.c\_str());

122 if (!DataFile)

123 {

124 std::cerr << "Data file with input not found \n";

125 return 0;

126 }

127

128 FILE \* pFile;

129 pFile=fopen((char\*)OutputFileName.c\_str(),"w");

130 fprintf(pFile, "PAR ANet AGross VPD Leaf\_Temp BoundaryL\_Conduc InternalCO2 Respiration StomatalConduct Transpiration\n");

131 OutputFile.open(OutputFileName.c\_str());

132 std::getline(ParamFile,DataLine); //get header from parameter file

133 // last line of file may be empty or contain numbers, this indicates file is at the end

134 while (!ParamFile.eof() && isalpha(CharTest)) //loops through file reads each line and tests if the correct species is found

135 {

136 getline(ParamFile,DataLine); // get the first line of data

137 CharTest=DataLine.at(0); //check that line contains alphabetical text at beginning

138 pnt=strtok\_s((char\*)DataLine.c\_str(), pDelim, &context ); // pick off the first word before the token (',')

139

140 //First read file to find the desired plant species

141 temp.assign(pnt);

142 temp.erase(remove(temp.begin(),temp.end(),' '),temp.end()); // removes spaces from string

143 thisParms.ID=temp;

144 temp.clear();

145

146

147

148 //Eliminate case issues - convert everything to lower case

149 transform(thisParms.ID.begin(), thisParms.ID.end(),thisParms.ID.begin(), ::tolower);

150 transform(Species.begin(), Species.end(),Species.begin(), ::tolower);

151

152 //Search for the correct species in file

153 if (thisParms.ID.compare(Species)==0 && !found) //continue to parse string

154 {

155

156 pnt = strtok\_s( NULL,pDelim, &context ); // iterate to clean string of characters already read

157 // This section parses string, each iteration it cleans string of characters already read

158 while(pnt!=NULL )

159 {

160 // printf( "Tokenized string using \* is:: %s\n", pnt ); // for debugging

161

162 temp.assign(pnt);

163 thisParms.species=temp;

164 temp.clear();

165 pnt = strtok\_s( NULL,pDelim, &context );

166 temp.assign(pnt);

167 thisParms.Type=temp;

168 temp.clear();

169 pnt = strtok\_s( NULL,pDelim, &context );

170 thisParms.Vcm25=atof(pnt);

171 pnt = strtok\_s( NULL,pDelim, &context );

172 thisParms.Jm25=atof(pnt);

173 pnt = strtok\_s( NULL,pDelim, &context );

174 thisParms.Vpm25=atof(pnt);

175 pnt = strtok\_s( NULL,pDelim, &context );

176 thisParms.TPU25=atof(pnt);

177 pnt = strtok\_s( NULL,pDelim, &context );

178 thisParms.Rd25=atof(pnt);

179 pnt = strtok\_s( NULL,pDelim, &context );

180 thisParms.Theta=atof(pnt);

181 pnt = strtok\_s( NULL,pDelim, &context );

182 thisParms.EaVc=atof(pnt);

183 pnt = strtok\_s( NULL,pDelim, &context );

184 thisParms.Eaj=atof(pnt);

185 pnt = strtok\_s( NULL,pDelim, &context );

186 thisParms.Hj=atof(pnt);

187 pnt = strtok\_s( NULL,pDelim, &context );

188 thisParms.Sj=atof(pnt);

189 pnt = strtok\_s( NULL,pDelim, &context );

190 thisParms.Hv=atof(pnt);

191 pnt = strtok\_s( NULL,pDelim, &context );

192 thisParms.EaVp=atof(pnt);

193 pnt = strtok\_s( NULL,pDelim, &context );

194 thisParms.Sv=atof(pnt);

195 pnt = strtok\_s( NULL,pDelim, &context );

196 thisParms.Eap=atof(pnt);

197 pnt = strtok\_s( NULL,pDelim, &context );

198 thisParms.Ear=atof(pnt);

199 pnt = strtok\_s( NULL,pDelim, &context );

200 thisParms.g0=atof(pnt);

201 pnt = strtok\_s( NULL,pDelim, &context );

202 thisParms.g1=atof(pnt);

203 pnt = strtok\_s( NULL,pDelim, &context );

204 thisParms.stomaRatio=atof(pnt);

205 pnt = strtok\_s( NULL,pDelim, &context );

206 thisParms.LfWidth=atof(pnt);

207 pnt = strtok\_s( NULL,pDelim, &context );

208 thisParms.LfAngFact=atof(pnt);

209 pnt = strtok\_s( NULL,pDelim, &context );

210 temp.assign(pnt);

211 thisParms.Remark=temp;

212 temp.clear();

213 pnt=strtok\_s(NULL,pDelim, &context);

214 found=true;

215 }

216

217 }

218 }

219 // Implementation to interact with photosynthesis model is here

220 CharTest='1'; //initialize CharTest

221 int start=1;

222 bool LineEmpty=false; //Checks if file with environmental data is finished.

223 //output variables

224

225 //Initialize Gas exchange object by passing species and relavent parameters read earlier

226 MyLeafGasEx->SetParams(&thisParms);

227 // loop to read environmental input data and call object to calculate results

228 while (!LineEmpty)

229

230 {

231 getline(DataFile, DataLine);

232 if (DataLine.length()==0) LineEmpty=true;

233

234 else

235 {

236

237 //CharTest=DataLine.at(0); // check for valid data

238 pnt=strtok\_s((char\*)DataLine.c\_str(), pDelim, &context ); //token is the delimiter

239 PFD=atof(pnt);

240 pnt=strtok\_s(NULL,pDelim,&context);

241 Temperature=atof(pnt);

242 pnt=strtok\_s(NULL,pDelim,&context);

243 CO2=atof(pnt);

244 pnt=strtok\_s(NULL,pDelim,&context);

245 RelativeHumidity=atof(pnt);

246 pnt=strtok\_s(NULL,pDelim,&context);

247 Wind=atof(pnt);

248 pnt=strtok\_s(NULL,pDelim,&context);

249 ConstantTemperature=atoi(pnt);

250 // pass relavent environmental variables to gas exchange object and execute module

251 MyLeafGasEx->SetVal(PFD, Temperature, CO2, RelativeHumidity,

252 Wind, Pressure, ConstantTemperature);

253 // Return calculated variables from gas exchange object

254 Anet=MyLeafGasEx->get\_ANet();

255 Agross=MyLeafGasEx->get\_AGross();

256 VPD=MyLeafGasEx->get\_VPD();

257 LeafTemperature=MyLeafGasEx->get\_LeafTemperature();

258 BoundaryLayerConductance=MyLeafGasEx->get\_BoundaryLayerConductance();

259 InternalCO2=MyLeafGasEx->get\_Ci();

260 Respiration=MyLeafGasEx->get\_Respiration();

261 StomatalConductance=MyLeafGasEx->get\_StomatalConductance();

262 Transpiration=MyLeafGasEx->get\_Transpiration();

263

264

265

266 fprintf(pFile,"%8.2f %6.2f %6.2f %8.3f %4.1f %8.3f %4.1f %6.2f %8.3f %8.3f\n", PFD, Anet, Agross, VPD,

267 LeafTemperature, BoundaryLayerConductance,InternalCO2,Respiration,

268 StomatalConductance, Transpiration);

269 }

270 }

271 DataFile.close();

272 DataFile.close();

273 fclose(pFile);

274 return 0;

275 }

## stdafx.cpp File Reference

#include "stdafx.h"

## stdafx.h File Reference

#include "targetver.h"

#include <stdio.h>

#include <tchar.h>

#include <atlbase.h>

#include <atlstr.h>

#include <string>

### Macros

* #define **\_ATL\_CSTRING\_EXPLICIT\_CONSTRUCTORS**

### Macro Definition Documentation

#### #define \_ATL\_CSTRING\_EXPLICIT\_CONSTRUCTORS

Definition at line 14 of file stdafx.h.

## targetver.h File Reference

### Macros

* #define **WINVER**  0x0600
* #define **\_WIN32\_WINNT**  0x0600
* #define **\_WIN32\_WINDOWS**  0x0410
* #define **\_WIN32\_IE**  0x0700

### Macro Definition Documentation

#### #define \_WIN32\_IE  0x0700

Definition at line 23 of file targetver.h.

#### #define \_WIN32\_WINDOWS  0x0410

Definition at line 19 of file targetver.h.

#### #define \_WIN32\_WINNT  0x0600

Definition at line 15 of file targetver.h.

#### #define WINVER  0x0600

Definition at line 11 of file targetver.h.

# Index

\_ATL\_CSTRING\_EXPLICIT\_CONSTRUCTORS

stdafx.h, 36

\_tmain

Photosynthesis(notManaged).cpp, 32

\_WIN32\_IE

targetver.h, 37

\_WIN32\_WINDOWS

targetver.h, 37

\_WIN32\_WINNT

targetver.h, 37

~CGasExchange

photomod::CGasExchange, 8

AssimilationGross

photomod::CGasExchange, 22

AssimilationNet

photomod::CGasExchange, 22

BoundaryLayerConductance

photomod::CGasExchange, 22

CalcStomatalConductance

photomod::CGasExchange, 8

CalcTurbulentVaporConductance

photomod::CGasExchange, 9

CGasExchange

photomod::CGasExchange, 8

Ci

photomod::CGasExchange, 22

Ci\_Ca

photomod::CGasExchange, 22

CO2

photomod::CGasExchange, 22

ConstantLeafTemperature

photomod::CGasExchange, 22

DarkRespiration

photomod::CGasExchange, 23

Eaj

photomod::CGasExchange::tParms, 27

Eap

photomod::CGasExchange::tParms, 27

Ear

photomod::CGasExchange::tParms, 27

EaVc

photomod::CGasExchange::tParms, 27

EaVp

photomod::CGasExchange::tParms, 27

EnergyBalance

photomod::CGasExchange, 10

epsilon

gas\_exchange.cpp, 30

eqlTolerance

photomod::CGasExchange, 23

errTolerance

photomod::CGasExchange, 23

Es

photomod::CGasExchange, 11

EvalCi

photomod::CGasExchange, 11

f

gas\_exchange.cpp, 30

g0

photomod::CGasExchange::tParms, 27

g1

photomod::CGasExchange::tParms, 28

gas\_exchange.cpp, 29

epsilon, 30

f, 30

maxiter, 30

O, 31

Q10, 31

R, 31

sbc, 31

scatt, 31

gas\_exchange.h, 31

GasEx

photomod::CGasExchange, 12

get\_AGross

photomod::CGasExchange, 13

get\_ANet

photomod::CGasExchange, 13

get\_BoundaryLayerConductance

photomod::CGasExchange, 13

get\_Ci

photomod::CGasExchange, 13

get\_CiCaRatio

photomod::CGasExchange, 13

get\_LeafTemperature

photomod::CGasExchange, 13

get\_Respiration

photomod::CGasExchange, 13

get\_StomatalConductance

photomod::CGasExchange, 13

get\_Transpiration

photomod::CGasExchange, 14

get\_VPD

photomod::CGasExchange, 14

Hj

photomod::CGasExchange::tParms, 28

Hv

photomod::CGasExchange::tParms, 28

ID

photomod::CGasExchange::tParms, 28

isCiConverged

photomod::CGasExchange, 23

iter\_Ci

photomod::CGasExchange, 23

iter\_total

photomod::CGasExchange, 23

iter1

photomod::CGasExchange, 23

iter2

photomod::CGasExchange, 23

Jm25

photomod::CGasExchange::tParms, 28

LfAngFact

photomod::CGasExchange::tParms, 28

LfWidth

photomod::CGasExchange::tParms, 28

maxiter

gas\_exchange.cpp, 30

Min

photomod::CGasExchange, 14

minh

photomod::CGasExchange, 14

O

gas\_exchange.cpp, 31

PhotoFluxDensity

photomod::CGasExchange, 23

photomod, 4

photomod::CGasExchange, 5

~CGasExchange, 8

AssimilationGross, 22

AssimilationNet, 22

BoundaryLayerConductance, 22

CalcStomatalConductance, 8

CalcTurbulentVaporConductance, 9

CGasExchange, 8

Ci, 22

Ci\_Ca, 22

CO2, 22

ConstantLeafTemperature, 22

DarkRespiration, 23

EnergyBalance, 10

eqlTolerance, 23

errTolerance, 23

Es, 11

EvalCi, 11

GasEx, 12

get\_AGross, 13

get\_ANet, 13

get\_BoundaryLayerConductance, 13

get\_Ci, 13

get\_CiCaRatio, 13

get\_LeafTemperature, 13

get\_Respiration, 13

get\_StomatalConductance, 13

get\_Transpiration, 14

get\_VPD, 14

isCiConverged, 23

iter\_Ci, 23

iter\_total, 23

iter1, 23

iter2, 23

Min, 14

minh, 14

PhotoFluxDensity, 23

PhotosynthesisC3, 15

PhotosynthesisC4, 16

PlantType, 24

Press, 24

QuadSolnLower, 18

QuadSolnUpper, 18

R\_abs, 24

RH, 24

SearchCi, 18

SetParams, 20

SetVal, 21

Slope, 21

sParms, 24

Square, 22

StomatalConductance, 24

Tair, 24

Tleaf, 24

Transpiration, 25

VPD, 25

width, 25

wind, 25

photomod::CGasExchange::tParms, 26

Eaj, 27

Eap, 27

Ear, 27

EaVc, 27

EaVp, 27

g0, 27

g1, 28

Hj, 28

Hv, 28

ID, 28

Jm25, 28

LfAngFact, 28

LfWidth, 28

Rd25, 28

Remark, 28

Sj, 28

species, 28

stomaRatio, 29

Sv, 29

Theta, 29

TPU25, 29

Type, 29

Vcm25, 29

Vpm25, 29

Photosynthesis(notManaged).cpp, 32

\_tmain, 32

PhotosynthesisC3

photomod::CGasExchange, 15

PhotosynthesisC4

photomod::CGasExchange, 16

PlantType

photomod::CGasExchange, 24

Press

photomod::CGasExchange, 24

Q10

gas\_exchange.cpp, 31

QuadSolnLower

photomod::CGasExchange, 18

QuadSolnUpper

photomod::CGasExchange, 18

R

gas\_exchange.cpp, 31

R\_abs

photomod::CGasExchange, 24

Rd25

photomod::CGasExchange::tParms, 28

Remark

photomod::CGasExchange::tParms, 28

RH

photomod::CGasExchange, 24

sbc

gas\_exchange.cpp, 31

scatt

gas\_exchange.cpp, 31

SearchCi

photomod::CGasExchange, 18

SetParams

photomod::CGasExchange, 20

SetVal

photomod::CGasExchange, 21

Sj

photomod::CGasExchange::tParms, 28

Slope

photomod::CGasExchange, 21

sParms

photomod::CGasExchange, 24

species

photomod::CGasExchange::tParms, 28

Square

photomod::CGasExchange, 22

stdafx.cpp, 36

stdafx.h, 36

\_ATL\_CSTRING\_EXPLICIT\_CONSTRUCTORS, 36

stomaRatio

photomod::CGasExchange::tParms, 29

StomatalConductance

photomod::CGasExchange, 24

Sv

photomod::CGasExchange::tParms, 29

Tair

photomod::CGasExchange, 24

targetver.h, 37

\_WIN32\_IE, 37

\_WIN32\_WINDOWS, 37

\_WIN32\_WINNT, 37

WINVER, 37

Theta

photomod::CGasExchange::tParms, 29

Tleaf

photomod::CGasExchange, 24

tparms, 25

TPU25

photomod::CGasExchange::tParms, 29

Transpiration

photomod::CGasExchange, 25

Type

photomod::CGasExchange::tParms, 29

Vcm25

photomod::CGasExchange::tParms, 29

VPD

photomod::CGasExchange, 25

Vpm25

photomod::CGasExchange::tParms, 29

width

photomod::CGasExchange, 25

wind

photomod::CGasExchange, 25

WINVER

targetver.h, 37