**Samara v2 Model**

()

**Module n°1 - RS\_InitParcelle\_V2**

This module initiates all relevant state variables for plot properties, namely hydrology, to their initial values at the beginning of the simulation. This can (should !) be way before the sowing date in order to let the soil water status establish itself according to weather conditions.

**1 - StockIniSurf** -IN- (en mm) : Stock d'eau initial dans l'horizon de surface

**2 - StockIniProf** -IN- (en mm) : Stock d'eau initial dans l'horizon de profondeur

**3 - EpaisseurSurf** -IN- (en mm) : Epaisseur de l'horizon de surface

**4 - EpaisseurProf** -IN- (en mm) : Epaisseur de l'horizon de profondeur

**5 - HumPF** -IN- (en m3/m3) : Humidité volumique au point de flétrissement (pF4.2)

**6 - HumFC** -IN- (en m3/m3)

**7 - HumSat** -IN- (en m3/m3) : Stock d'eau à la saturation

**8 - PEvap** -IN- (en Coeff x) : Seuil d'évaporation au régime potentiel.

**9 - DateSemis** -IN- (en Date) : Date de semis

**10 - ResUtil** -OUT- (en mm/m)

**11 - StockTotal** -OUT- (en mm) : Total water column stored in soil profile

**12 - LTRkdfcl** -OUT- (en fraction) : Light transmission rate of canopy as calculated with Kdfcl (taking into account crop Kdf and clumping), = 1-LIRkdfcl

**13 - Hum** -OUT- (en mm) : Quantité d'eau maximum jusqu'au front d'humectation

**14 - RuSurf** -OUT- (en mm) : Reserve utile de l'horizon de surface

**15 - ProfRu** -OUT- (en mm) : Profondeur maximale de sol

**16 - StRuMax** -OUT- (en mm) : Capacité maximale de la RU

**17 - CapaREvap** -OUT- (en mm) : Capacité du réservoir d'évaporation

**18 - CapaRFE** -OUT- (en mm) : Capacité du réservoir facilement évaporable (au potentiel)

**19 - CapaRDE** -OUT- (en mm) : Réserve difficilement transpirable mais évaporable

**20 - ValRSurf** -OUT- (en mm) : Contenu des 2 réservoirs RDE et REvap

**21 - ValRDE** -OUT- (en mm) : Contenu de la RDE

**22 - ValRFE** -OUT- (en mm) : Contenu de la RFE

**23 - StockSurface** -OUT- (en mm) : Water column stored in topsoil layer

**24 - CounterNursery** -OUT-

**25 - VolRelMacropores** -OUT- (en %) : Rel. Volume of macropores in soil (%) = air spaces that are filled with air when soil saturated but freely drained

**26 - VolMacropores** -OUT-

procedure RS\_InitParcelle\_V2(const StockIniSurf, StockIniProf, EpaisseurSurf, EpaisseurProf, HumPF, HumFC, HumSat, PEvap, DateSemis : Double; var ResUtil, StockTotal, LTRkdfcl, Hum, RuSurf, ProfRU, StRuMax, CapaREvap, CapaRFE, CapaRDE, ValRSurf, ValRDE, ValRFE, StockSurface, CounterNursery, VolRelMacropores, VolMacropores : Double);

var

Stockini2 : Double;

Stockini1 : Double;

begin

try

VolRelMacropores := 100 \* (HumSat - HumFC);

ResUtil := (HumFC - HumPF) \* 1000;

ProfRU := EpaisseurSurf + EpaisseurProf ; // à supprimer ultérieurement dans les différents modules

RuSurf := ResUtil \* EpaisseurSurf / 1000;

CapaREvap := 0.5 \* EpaisseurSurf \* HumPF;

CapaRFE := PEvap \* (CapaREvap + RuSurf);

CapaRDE := RuSurf - CapaRFE;

StRuMax := ResUtil \* ProfRu / 1000;

Stockini1 := Min(StockIniSurf, CapaREvap + RuSurf);

Stockini2 := Min(StockIniProf, ResUtil \* EpaisseurProf / 1000);

ValRSurf := Min(Stockini1, CapaREvap + CapaRDE);

ValRDE := Max(0, ValRSurf - CapaREvap);

ValRFE := Max(0, Stockini1 - (CapaREvap + CapaRDE));

StockSurface := ValRDE + ValRFE ;

StockTotal := StockSurface + Stockini2;//transpirable

Hum := StockTotal ;

LTRkdfcl := 1;

CounterNursery := 0;

VolMacropores := VolRelMacropores \* (EpaisseurSurf + EpaisseurProf) / 100;

except

AfficheMessageErreur('RS\_InitParcelle\_V2', URisocas);

end;

end;

**Module n°2 - RS\_InitiationCulture**

This module initiates all relevant state variables of the crop, namely phenology, to their initial values at the time of sowing.

**1 - SDJLevee** -IN- (en °C.d) : Phase 1. Sets duration from sowing to germination (but may be overrode by drought)

**2 - SDJBVP** -IN- (en °C.d) : Phase 2. Sets duration from germination to earliest possible PI (onset of BVP)

**3 - SDJRPR** -IN- (en °C.d) : Phase 4. Sets duration from PI to Flowering. Period of internode and panicle (structural component) development

**4 - SDJMatu1** -IN- (en °C.d) : Phase 5. Sets duration from flowering to end of grain filling. No more structural growth happens

**5 - SDJMatu2** -IN- (en °C.d) : Phase 6: Sets duration from end of grain filling to maturity/harvest date. No more growth but Assimilation & Rm continue, causing changes in IN

**6 - SommeDegresJourMax** -OUT- (en °C.jour) : Somme des degrés/jour pour le cycle de la plante

**7 - NumPhase** -OUT- (en none) : Phenological phase

**8 - SumDegresDay** -OUT- (en °C.jour) : Somme de degrés.jours depuis le début de la phase 1

**9 - SeuilTemp** -OUT- (en °C.jour) : Seuil des températures cumulées pour la phase en cours

**10 - Lai** -OUT- (en m²/m²) : leaf area index (green leaf blades only)

**11 - IcCum** -OUT- (en kg/kg)

**12 - FTSW** -OUT- (en none) : fraction of transpirable soil water within the bulk root zone

**13 - Cstr** -OUT- (en none) : drought stress coefficient: FTSW is transformed into Cstr by FAO function using P-factor

procedure RS\_InitiationCulture(Const SeuilTempLevee, SeuilTempBVP, SeuilTempRPR, SeuilTempMatu1, SeuilTempMatu2 : Double; var SommeDegresJourMaximale, NumPhase, SommeDegresJour, SeuilTempPhaseSuivante, Lai, IcCumul, FTSW, cstr : Double);

begin

try

NumPhase := 0;

SommeDegresJourMaximale := SeuilTempLevee + SeuilTempBVP + SeuilTempRPR + SeuilTempMatu1 + SeuilTempMatu2;

SommeDegresJour := 0;

SeuilTempPhaseSuivante := 0;

Lai := 0;

IcCumul := 0;

FTSW := 1;

cstr := 1;

except

AfficheMessageErreur('RS\_InitiationCulture',URisocas);

end;

end;

**Module n°3 - RS\_Transplanting\_V2**

This module manages, for the case of bunded lowland conditions, the transplanting of the crop. This is only done of the binary cultural practices parameter “Transplanting” is =1. The model simulates the growth of the crop initially in the seedbed nursery at the density “DensityNursery” (parameter) until the period of “DurationNursery” (parameter) is over. The binary state variable “NurseryStatus” checks whether the crop is still in the nursery. Upon transplanting to the final population density (DensityField, parameter), the simulated stand is a lot thinner and therefore, LAI and dry matter state variables go down. Recovery from transplanting shock can be immediate or may involve 10 days of reduced photosynthesis, depending on the choice of value for “TansplantingShock” (parameter). Attention: Crop output state variables, if on a per-area (XXXpop) basis, are calculated the same way in the nursery and in the field (kg/ha), thus the biomass simulated for the nursery may be misleading (calculation is kg/ha, but the nursery is usually just a few sqm). Water relations and management are simulated the same way in the nursery and in the field. Note that the simulation fully takes into account the high level of competition in the nursery, resulting in small biomass per plant, which then recovers in the main field. The optional parameter setting “LifeSavingDrainage = 1” helps avoiding submergence in the nursery and in the main field alike. If parameter setting “AutoIrrig = 1” is selected, bund height is automatically adjusted daily to ensure that floodwater depth is kept at 50% plant height.

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - DensityNursery** -IN- (en pieds/Ha)

**3 - DensityField** -IN- (en pieds/Ha)

**4 - DurationNursery** -IN- (en d) : Time from Sowing to transplanting

**5 - PlantsPerHill** -IN- : Number of seeds placed together in a hill (supposing all will germinate and grow)

**6 - Transplanting** -IN- (en none) : If value=1 then crop is grown in seedling nursery for (DurationNursery) days, the transplanted at the population density set by the other params

**7 - NurseryStatus** -INOUT-

**8 - ChangeNurseryStatus** -INOUT-

**9 - CounterNursery** -INOUT-

**10 - Density** -INOUT- (en pieds/Ha)

**11 - DryMatStructLeafPop** -INOUT- (en kg/ha) : Green leaf blade dry matter at population scale

**12 - DryMatStructSheathPop** -INOUT- (en kg/ha) : Sheath blade dry matter at population scale

**13 - DryMatStructRootPop** -INOUT- (en kg/ha) : Root blade dry matter at population scale

**14 - DryMatStructInternodePop** -INOUT- (en kg/ha) : Internode blade dry matter at population scale (only structural component: reserves are simulated and output separately)

**15 - DryMatStructPaniclePop** -INOUT- (en kg/ha) : Panicle structural dry matter at population scale (does not include grains), formed between PI and flowering

**16 - DryMatResInternodePop** -INOUT-

procedure RS\_Transplanting\_V2(const NumPhase, DensityNursery, DensityField, DurationNursery, PlantsPerHill, Transplanting : Double; var NurseryStatus, ChangeNurseryStatus, CounterNursery, Density, DryMatStructLeafPop, DryMatStructSheathPop, DryMatStructRootPop, DryMatStructInternodePop, DryMatStructPaniclePop, DryMatResInternodePop : Double);

var

DensityChange : Double;

begin

try

DensityChange := DensityField / (DensityNursery / PlantsPerHill);

if ((Transplanting = 1) and (NumPhase >= 1)) then

begin

CounterNursery := CounterNursery + 1;

end;

if ((Transplanting = 1) and (CounterNursery < DurationNursery) and (ChangeNurseryStatus = 0)) then

begin

NurseryStatus := 0;

ChangeNurseryStatus := 0;

end

else

begin

if ((Transplanting = 1) and (CounterNursery >= DurationNursery) and (ChangeNurseryStatus = 0) and (NurseryStatus = 0)) then

begin

NurseryStatus := 1;

ChangeNurseryStatus := 1;

end

else

begin

NurseryStatus := 1;

ChangeNurseryStatus := 0;

end;

end;

if (NurseryStatus = 1) then

begin

Density := DensityField;

end

else

begin

Density := DensityNursery / PlantsPerHill;

end;

if (ChangeNurseryStatus = 1) then

begin

DryMatStructLeafPop := DryMatStructLeafPop \* DensityChange;

DryMatStructSheathPop := DryMatStructSheathPop \* DensityChange;

DryMatStructRootPop := DryMatStructRootPop \* DensityChange;

DryMatStructInternodePop := DryMatStructInternodePop \* DensityChange;

DryMatStructPaniclePop := DryMatStructPaniclePop \* DensityChange;

DryMatResInternodePop := DryMatResInternodePop \* DensityChange;

end;

except

AfficheMessageErreur('RS\_Transplanting\_V2', URisocas);

end;

end;

**Module n°4 - Meteo0DegToRad**

This module converts Deg latitude to Rad latitude for photoperiod calculation.

**1 - Latitude** -IN- (en °) : Latitude

**2 - LatRad** -OUT- (en radian) : Latitude en radians

procedure DegToRad(const Lat : Double; var LatRad : Double);

begin

try

LatRad := Lat \* PI /180;

except

AfficheMessageErreur('DegToRad',UMeteo);

end;

end;

**Module n°5 - Meteo1AVGTempHum**

This module mean T and humidity from the min and max.

**1 - TMin** -IN- (en °C) : Température minimale mesurée

**2 - TMax** -IN- (en °C) : Température maximale mesurée

**3 - HMin** -IN- (en %) : Humidité minimale mesurée

**4 - HMax** -IN- (en %) : Humidité maximale mesurée

**5 - TMoy** -IN- (en °C) : Température moyenne mesurée

**6 - HMoy** -IN- (en %) : Humidité moyenne mesurée

**7 - TMoyCalc** -OUT- (en °C) : Mean of Tmin and Tmax

**8 - HMoyCalc** -OUT- (en %) : Mean of min and max humidity

procedure AVGTempHum(const TMin, TMax, HMin, HMax, TMoy, HMoy : Double; var TMoyCalc, HMoyCalc : Double);

begin

try

if ((TMin <> NullValue) and (TMax <> NullValue)) then

begin

TMoyCalc := (TMax + TMin) / 2;

end

else

begin

TMoyCalc := TMoy;

end;

if ((HMin <> NullValue) and (HMax <> NullValue)) then

begin

HMoyCalc := (HMax + HMin) / 2;

end

else

begin

HMoyCalc := HMoy;

end;

except

AfficheMessageErreur('AVGTempHum',UMeteo);

end;

end;

**Module n°6 - Meteo2Decli**

**1 - DateEnCours** -IN- (en Date) : Date du pas de simulation en cours

**2 - Decli** -OUT- (en radian) : Declinaison du soleil

procedure EvalDecli(const aDate : TDateTime; var Decli : Double);

begin

try

Decli := 0.409 \* Sin(0.0172 \* DayOfTheYear(aDate) - 1.39);

except

AfficheMessageErreur('EvalDecli',UMeteo);

end;

end;

**Module n°7 - Meteo3SunPosi**

This module calculates the sun position according to latitude and season.

**1 - LatRad** -IN- (en radian) : Latitude en radians

**2 - Decli** -IN- (en radian) : Declinaison du soleil

**3 - SunPosi** -OUT- : Position du soleil

procedure EvalSunPosi(const LatRad, Decli : Double; var SunPosi : Double);

begin

try

SunPosi := Arccos(-Tan(LatRad) \* Tan(Decli));

except

AfficheMessageErreur('EvalSunPosi',UMeteo);

end;

end;

**Module n°8 - Meteo4DayLength**

This module calculates Day Length.

**1 - SunPosi** -IN- : Position du soleil

**2 - DayLength** -OUT- (en hour(dec)) : day length including civil twilight

procedure EvalDayLength(const SunPosi : Double; var DayLength : Double);

begin

try

DayLength := 7.64 \* SunPosi;

except

AfficheMessageErreur('EvalDayLength',UMeteo);

end;

end;

**Module n°9 - Meteo5SunDistance**

This module calculates SunDistance, needed for the calculation of extraterrestrial radiation, needed for global radiation calculation from sunshine hours (where Rs data are not available).

**1 - DateEnCours** -IN- (en Date) : Date du pas de simulation en cours

**2 - SunDistance** -OUT- : Distance relative du soleil à la terre

procedure EvalSunDistance(const aDate : TDatetime;var SunDistance: Double);

begin

try

SunDistance := 1 + 0.033 \* Cos(2 \* PI / 365 \* DayOfTheYear(aDate));

except

AfficheMessageErreur('EvalSunDistance',UMeteo);

end;

end;

**Module n°10 - Meteo6RayExtra**

This module calculates extraterrestrial radiation, needed for global radiation calculation from sunshine hours (where Rs data are not available).

**1 - SunPosi** -IN- : Position du soleil

**2 - Decli** -IN- (en radian) : Declinaison du soleil

**3 - SunDistance** -IN- : Distance relative du soleil à la terre

**4 - LatRad** -IN- (en radian) : Latitude en radians

**5 - RayExtra** -OUT- (en MJ/m²/d) : Extra-terrestrial solar radiation

procedure EvalRayExtra(const SunPosi, Decli, SunDistance, LatRad : Double; var RayExtra : Double);

begin

try

RayExtra := 24 \* 60 \* 0.0820 / PI \* SunDistance \*

(SunPosi \* Sin(Decli) \* Sin(LatRad) +

Cos(Decli) \* Cos(LatRad) \* Sin(SunPosi));

except

AfficheMessageErreur('EvalRayExtra',UMeteo);

end;

end;

**Module n°11 - Meteo7RgMax**

This module calculates maximal radiation at ground level, needed for global radiation calculation from sunshine hours (where Rs data are not available).

**1 - RayExtra** -IN- (en MJ/m²/d) : Extra-terrestrial solar radiation

**2 - Altitude** -IN- (en m) : Altitude du site

**3 - RgMax** -OUT- (en MJ/m²/d) : Rayonnement global maximum du jour si ciel clair

procedure EvalRgMax(const RayExtra, Alt : Double; var RgMax : Double);

begin

try

RgMax := (0.75 + 0.00002 \* Alt) \* RayExtra ;

except

AfficheMessageErreur('EvalRgMax',UMeteo);

end;

end;

**Module n°12 - Meteo8InsToRg**

This module calculates global radiation from sunshine hours (where Rs data are not available).

**1 - DayLength** -IN- (en hour(dec)) : day length including civil twilight

**2 - Ins** -IN- (en none) : Durée d’insolation

**3 - RayExtra** -IN- (en MJ/m²/d) : Extra-terrestrial solar radiation

**4 - RgMax** -IN- (en none) : Rayonnement global maximum du jour si ciel clair

**5 - Rg** -IN- (en none) : Rayonnement global

**6 - RgCalc** -OUT- (en MJ/m²/d) : Solar global radiation as calculated from sunshine hours, calendar date and latitude for cases of unavailability of direct measurements of Rg

procedure InsToRg(const DayLength, Ins, RayExtra, RgMax,RGLue : Double; var RGCalc : Double);

begin

try

if (RGLue = NullValue) then

begin

RGCalc := (0.25 + 0.50 \* Min(Ins / DayLength, 1)) \* RayExtra;

end

else

begin

RGCalc := RGLue;

end;

except

AfficheMessageErreur('InsToRg',UMeteo);

end;

end;

**Module n°13 - Meteo9Par**

This module calculates PAR from global radiation.

**1 - RgCalc** -IN- (en MJ/m²/d) : Solar global radiation as calculated from sunshine hours, calendar date and latitude for cases of unavailability of direct measurements of Rg

**2 - KPar** -IN- (en MJ/MJ) : Coeff de conversion du RG en Par (part de rayonnement photosynthétiquement actif)

**3 - Par** -OUT- (en MJ/m²/d) : Photosynthetically active radiation (PAR), which is about 50% of incoming global solar radiation

procedure EvalPar(const RG, KPar : Double; var Par : Double);

begin

try

Par := KPar \* Rg;

except

AfficheMessageErreur('EvalPar',UMeteo);

end;

end;

**Module n°14 - MeteoEToFAO**

This module calculates reference evapotranspiration from meteo data according to FAO standard. Needed to drive soil evaporation and plant transpiration.

**1 - ETP** -IN- (en mm)

**2 - Altitude** -IN- (en m) : Altitude du site

**3 - RgMax** -IN- (en none) : Rayonnement global maximum du jour si ciel clair

**4 - RgCalc** -IN- (en MJ/m²/d) : Solar global radiation as calculated from sunshine hours, calendar date and latitude for cases of unavailability of direct measurements of Rg

**5 - TMin** -IN- (en °C) : Température minimale mesurée

**6 - TMax** -IN- (en °C) : Température maximale mesurée

**7 - HMin** -IN- (en %) : Humidité minimale mesurée

**8 - HMax** -IN- (en %) : Humidité maximale mesurée

**9 - HMoyCalc** -IN- (en %) : Mean of min and max humidity

**10 - TMoyCalc** -IN- (en °C) : Mean of Tmin and Tmax

**11 - Vt** -IN- (en m/s) : Vitesse moyenne journalière du vent à 2 m

**12 - ETo** -OUT- (en mm/d) : potential evapotranspiration (FAO, also called PET, ETP or Eto). Approximates atmospheric demand for water vapor applied to a calm water surface

**13 - TMoyPrec** -INOUT- : Température moyenne du jour précédent

**14 - VDPCalc** -OUT- (en kgPa) : Vapor Pressure Deficit (VPD) calculated from relative humidity and temperature

procedure EToFAO(const ETP, Alt, RgMax, RayGlobal, TMin, TMax, HrMin, HrMax, HrMoy, Tmoy, Vent : Double; var ETo, TMoyPrec, VPD : Double);

var

eActual, eSat,

RgRgMax, TLat, delta, KPsy,

Eaero, Erad, Rn, G : Double;

begin

try

if (ETP = NullValue) then

begin

eSat := 0.3054 \* (Exp(17.27 \* TMax / (TMax + 237.3)) +

exp (17.27 \* TMin / (TMin + 237.3)));

if (HrMax = NullValue) then

eActual := eSat \* HrMoy / 100

else

eActual := 0.3054 \* (Exp(17.27 \* TMax / (TMax + 237.3)) \*

HrMin/100 + Exp(17.27 \* TMin / (TMin + 237.3)) \*

HrMax / 100);

VPD := eSat-eActual;

RgRgMax := RayGlobal / RgMax;

if (RgRgMax > 1) then

RgRgMax := 1;

Rn := 0.77 \* RayGlobal - (1.35 \* RgRgMax - 0.35) \*

(0.34 - 0.14 \* Power(eActual, 0.5)) \*

(Power(TMax + 273.16, 4) + Power(TMin + 273.16, 4)) \* 2.45015 \* Power(10, -9);

// chaleur latente de vaporisation de l'eau

Tlat := 2.501 - 2.361 \* power(10, -3) \* Tmoy;

// pente de la courbe de pression de vapeur saturante en kPa/°C

delta := 4098 \* (0.6108 \* Exp(17.27 \* Tmoy / (Tmoy + 237.3))) / Power(Tmoy + 237.3, 2);

// constante psychrométrique en kPa/°C

Kpsy := 0.00163 \* 101.3 \* power(1 - (0.0065 \* Alt / 293), 5.26) / TLat;

// Radiative

G := 0.38 \* (Tmoy - TmoyPrec);

Erad := 0.408 \* (Rn - G) \* delta / (delta + Kpsy \* ( 1 + 0.34 \* Vent));

// Partie évaporative de ET0 = Eaéro

Eaero := (900 / (Tmoy + 273.16)) \* ((eSat - eActual) \* Vent ) \* Kpsy /

(delta + Kpsy \* ( 1 + 0.34 \* Vent));

Eto := Erad + Eaero;

end

else

begin

Eto := ETP;

end;

TMoyPrec := TMoy;

except

AfficheMessageErreur('EToFAO',UMeteo);

end;

end;

**Module n°15 - RizPhenoPSPStress**

This module calculates the progress in crop phenology across the phases (state variable “NumPhase”) 0 (before sowing), 1 (sowing to germination), 2 (Basic Vegetative Phase BVP), 3 (Photoperiod sensitive Phase PSP ending with panicle initiation), 4 (Reproductivephase ending with flowering), 5 (Maturation phase 1 = grain filling), 6 (Maturation phase 2 = grain drying) and 7 (maturity, just one day, then end of crop cycle). The photoperiodic effect on duration of PSP (NumPhase = 3) is calculated according to the published “Impatience” model in **Module n°90 - RS\_EvolPSPMVMD.**

Note: this module needs improvement because it does not consider diurnal courses of T.

Cumul photopériodique depuis le début de la phase PSP

Modification pour gérer le module générique de photopériode de M. Vaksman et M. Dingkuhn

**1 - SumPP** -IN- (en °C.d) : Cumul photopériodique depuis le début de la phase PSP

**2 - PPSens** -IN- (en none) : PP sensitivity, important variable. Range 0.3-0.6 is PP sensitive, sensititivity disappears towards values of 0.7 to 1

**3 - SumDegreDayCor** -IN- (en °C.jour)

**4 - SDJLevee** -IN- (en °C.d) : Phase 1. Sets duration from sowing to germination (but may be overrode by drought)

**5 - SDJBVP** -IN- (en °C.d) : Phase 2. Sets duration from germination to earliest possible PI (onset of BVP)

**6 - SDJRPR** -IN- (en °C.d) : Phase 4. Sets duration from PI to Flowering. Period of internode and panicle (structural component) development

**7 - SDJMatu1** -IN- (en °C.d) : Phase 5. Sets duration from flowering to end of grain filling. No more structural growth happens

**8 - SDJMatu2** -IN- (en °C.d) : Phase 6: Sets duration from end of grain filling to maturity/harvest date. No more growth but Assimilation & Rm continue, causing changes in IN

**9 - StockSurface** -IN- (en mm) : Water column stored in topsoil layer

**10 - TxRuSurfGermi** -IN- (en Coeff x) : Sets top soil relative water content necessary to enable germination

**11 - RuSurf** -IN- (en mm) : Reserve utile de l'horizon de surface

**12 - DateEnCours** -IN- (en Date) : Date du pas de simulation en cours

**13 - DateSemis** -IN- (en Date) : Date de semis

**14 - StockTotal** -IN- (en mm) : Total water column stored in soil profile

**15 - NumPhase** -INOUT- (en none) : Phenological phase

**16 - SumDDPhasePrec** -INOUT- (en °C.jour) : Somme en degrés/jour de la phase précédente

**17 - SeuilTemp** -INOUT- (en °C.jour) : Seuil des températures cumulées pour la phase en cours

**18 - ChangePhase** -INOUT- : ce booléen permet de savoir si la journée courante est une journée de changement de phase (facilite l'initialisation)

**19 - SeuilTempSsPhase** -INOUT- (en °C.jour)

**20 - ChangeSsPhase** -INOUT-

**21 - NumSsPhase** -INOUT-

procedure EvolPhenoPSPStress(const SumPP, PPsens, SommeDegresJour, SeuilTempLevee, SeuilTempBVP, SeuilTempRPR, SeuilTempMatu1, SeuilTempMatu2, StockSurface, PourcRuSurfGermi, RuSurf, DateDuJour, DateSemis, stRu : Double; var NumPhase, SommeDegresJourPhasePrec, SeuilTempPhaseSuivante, ChangePhase, SeuilTempSousPhaseSuivante, ChangeSousPhase, NumSousPhase : Double);

{Procedure speciale pour inclure le module photoperiodique de Vaksman & Dingkuhn

qui fonctionne en degres jours et declanche IP lorsque SumPP est inferieur à PPsens}

//JCC le 21/09/05

// Cette procédure est appelée en début de journée et fait évoluer les phase

// phénologiques. Pour celà, elle incrémente les numéro de phase et change la

// valeur du seuil de la phase suivante. ChangePhase est un booléen permettant

// d'informer le modèle pour connaître si un jour est un jour de changement

// de phase. Celà permets d'initialiser les variables directement dans les

// modules spécifiques.

// --> Stades phénologiques pour le modèle Mil réécrit:

// 0 : du jour de semis au début des conditions favorables pour la germination et de la récolte à la fin de simulation (pas de culture)

// 1 : du début des conditions favorables pour la germination au jour de la levée

// 2 : du jour de la levée au début de la phase photopériodique

// 3 : du début de la phase photopériodique au début de la phase reproductive

// 4 : du début de la phase reproductive au début de la maturation

// sousphase1 de début RPR à RPR/4

// sousphase2 de RPR/4 à RPR/2

// sousphase3 de RPR/2 à 3/4 RPR

// sousphase4 de 3/4 RPR à fin RPR

// 5 : du début de la maturation au début du séchage

// 6 : du début du séchage au jour de récolte

// 7 : le jour de la récolte

var

ChangementDePhase,

ChangementDeSousPhase : Boolean; // on passe en variable un pseudo booléen et non directement ce booléen (pb de moteur)

begin

try

ChangePhase := 0;

ChangeSousPhase := 0;

// l'initialisation quotidienne de cette variable à faux permet de stopper le marquage d'une journée de changement de phase

if (Trunc(NumPhase) = 0) then // la culture a été semée mais n'a pas germé

begin

if ((StockSurface >= PourcRuSurfGermi \* RuSurf) or (stRu > StockSurface)) then

begin // on commence ds les conditions favo aujourd'hui

NumPhase := 1;

ChangePhase := 1;

SeuilTempPhaseSuivante := SeuilTempLevee;

end;

end // fin du if NumPhase=0

else

begin

// vérification d'un éventuel changement de phase

if ((Trunc(NumPhase) = 1) and (SommeDegresJour >= SeuilTempPhaseSuivante)) then //si on change de phase de BVP à PSP aujourd'hui

ChangementDePhase := True

else

begin //sinon

if (Trunc(NumPhase) <> 3) Then

begin

ChangementDePhase := (SommeDegresJour >= SeuilTempPhaseSuivante);

end

else

begin

ChangementDePhase := (sumPP <= PPsens); // true=on quittera la phase photopériodique

end;

end;

// on a changé de phase

if ChangementDePhase then

begin

ChangePhase := 1;

NumPhase := NumPhase + 1;

SommeDegresJourPhasePrec := SeuilTempPhaseSuivante; // utilisé dans EvalConversion

case Trunc(NumPhase) of

2 : SeuilTempPhaseSuivante := SeuilTempPhaseSuivante + SeuilTempBVP; // BVP Developpement vegetatif

4 : begin

// gestion de l'initialisation des sous-phases

SeuilTempSousPhaseSuivante := SeuilTempPhaseSuivante + SeuilTempRPR / 4; // initialisation de la somme des DJ de la 1ère sous phase

NumSousPhase := 1; // initialisation du n° de sous phase

MonCompteur := 0; // on est bien le 1er jour de la 1ere sous phase

ChangeSousPhase := 1; // on est bien un jour de changement de sous phase (en locurence, la première...)

// gestion du seuil de la phase suivante

SeuilTempPhaseSuivante := SeuilTempPhaseSuivante + SeuilTempRPR; // RPR Stade initiation paniculaire

end;

5 : SeuilTempPhaseSuivante := SeuilTempPhaseSuivante + SeuilTempMatu1; // Matu1 remplissage grains

6 : SeuilTempPhaseSuivante := SeuilTempPhaseSuivante + SeuilTempMatu2; // Matu2 dessication

end; // Case NumPhase

end; // end change

// gestion des sous-phases de la phase RPR (4)

if (Trunc(NumPhase) = 4) then

begin

ChangementDeSousPhase := (SommeDegresJour >= SeuilTempSousPhaseSuivante);

if ChangementDeSousPhase then

begin

SeuilTempSousPhaseSuivante := SeuilTempSousPhaseSuivante + SeuilTempRPR / 4;

NumSousPhase := NumSousPhase + 1;

MonCompteur := 1;

ChangeSousPhase := 1;

end

else

Inc(MonCompteur);

end; // fin du if Trunc(NumPhase)=4 then

end;

except

AfficheMessageErreur('EvolPhenoStress | NumPhase: '+FloatToStr(NumPhase)+

' SommeDegresJour: '+FloatToStr(SommeDegresJour)+

' SeuilTempPhaseSuivante: '+FloatToStr(SeuilTempPhaseSuivante) ,URiz);

end;

end;

**Module n°16 - RS\_EvalSimAnthesis50**

This module calculates the days elapsing since germination, until maturity or end of crop simulation.

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - ChangePhase** -IN- : ce booléen permet de savoir si la journée courante est une journée de changement de phase (facilite l'initialisation)

**3 - NbJAS** -IN- (en d) : days after sowing

**4 - SimAnthesis50** -INOUT- (en d)

procedure RS\_EvalSimAnthesis50(const NumPhase, ChangePhase, NbJas : Double; var SimAnthesis50 : Double);

begin

try

if (NumPhase = 5) and (ChangePhase = 1) then

begin

SimAnthesis50 := NbJas

end;

except

AfficheMessageErreur('RS\_EvalSimAnthesis50', URisocas);

end;

end;

**Module n°17 - RS\_EvalDateGermination**

This module calculates the days elapsing since germination, until maturity or end of crop simulation.

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - ChangePhase** -IN- : ce booléen permet de savoir si la journée courante est une journée de changement de phase (facilite l'initialisation)

**3 - NbDaysSinceGermination** -INOUT-

procedure RS\_EvalDateGermination(const NumPhase, ChangePhase : Double; var NbDaysSinceGermination : double);

begin

try

if ((NumPhase < 1) or ((NumPhase = 1) and (ChangePhase = 1))) then

begin

NbDaysSinceGermination := 0;

end

else

begin

NbDaysSinceGermination := NbDaysSinceGermination + 1;

end;

except

AfficheMessageErreur('RS\_EvalDateGermination', URisocas);

end;

end;

**Module n°18 - RS\_EvalColdStress**

This module provides the possibility to introduce a cold stress (daily min T) effect on development rate (reduction of effective thermal time of the day), associated with a reduction in A (supposed to be less sensitive than development rate, using a non linear function). Whernever daily Tmin drops to within the interval between **KCritStressCold1** and **KCritStressCold2**, or below, there is a proportional slowing of development and a non-linear reduction in A. This comes in addition to the thermal time effect. Such cold stress effects have been obserbed in the Sahel (Sabine Stürz’ thesis). Difficult to distinguish from PP effects, but identifiable by stunting and leaf death in the field, associated with an increase in crop duration.

**1 - KCritStressCold1** -IN- (en °C) : Upper critical Tmin for triggering development delay

**2 - KCritStressCold2** -IN- (en °C) : Lower critical Tmin triggering development delay

**3 - TMin** -IN- (en °C) : Température minimale mesurée

**4 - StressCold** -OUT- (en Coeff x)

procedure RS\_EvalColdStress(const KCritStressCold1, KCritStressCold2, TMin : Double; var StressCold : Double);

begin

try

StressCold := 1 - Max(0, Min(1, KCritStressCold1 / (KCritStressCold1 - KCritStressCold2) - TMin / (KCritStressCold1 - KCritStressCold2)));

StressCold := Max(0.00001, StressCold);

except

AfficheMessageErreur('RS\_EvalColdStress', URisocas);

end;

end;

**Module n°19 - RS\_EvalSimEmergence**

This modules identifies the days after sowing when emergence happens (start of growth)

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - ChangePhase** -IN- : ce booléen permet de savoir si la journée courante est une journée de changement de phase (facilite l'initialisation)

**3 - NbJAS** -IN- (en d) : days after sowing

**4 - SimEmergence** -INOUT- (en d)

procedure RS\_EvalSimEmergence(const NumPhase, ChangePhase, NbJas : Double; var SimEmergence : Double);

begin

try

if (NumPhase = 2) and (ChangePhase = 1) then

begin

SimEmergence := NbJas

end;

except

AfficheMessageErreur('RS\_EvalSimEmergence', URisocas);

end;

end;

**Module n°20 - RS\_EvalSimPanIni**

This modules identifies the days after sowing when panicle initiation happens

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - ChangePhase** -IN- : ce booléen permet de savoir si la journée courante est une journée de changement de phase (facilite l'initialisation)

**3 - NbJAS** -IN- (en d) : days after sowing

**4 - SimPanIni** -INOUT- (en d)

procedure RS\_EvalSimPanIni(const NumPhase, ChangePhase, NbJas : Double; var SimPanIni : Double);

begin

try

if (NumPhase = 4) and (ChangePhase = 1) then

begin

SimPanIni := NbJas

end;

except

AfficheMessageErreur('RS\_EvalSimPanIni', URisocas);

end;

end;

**Module n°21 - RS\_EvalSimStartGermin**

This modules identifies the days after sowing when germination starts (no growth simulated at this point). This may not be identical to sowing date because soil wetting may be insufficient.

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - ChangePhase** -IN- : ce booléen permet de savoir si la journée courante est une journée de changement de phase (facilite l'initialisation)

**3 - NbJAS** -IN- (en d) : days after sowing

**4 - SimStartGermin** -INOUT- (en d)

procedure RS\_EvalSimStartGermin(const NumPhase, ChangePhase, NbJas : Double; var SimStartGermin : Double);

begin

try

if (NumPhase = 1) and (ChangePhase = 1) then

begin

SimStartGermin := NbJas

end;

except

AfficheMessageErreur('RS\_EvalSimStartGermin', URisocas);

end;

end;

**Module n°22 - RS\_EvalSimStartMatu2**

This modules identifies the days after sowing when grain filling ends and grains dry up

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - ChangePhase** -IN- : ce booléen permet de savoir si la journée courante est une journée de changement de phase (facilite l'initialisation)

**3 - NbJAS** -IN- (en d) : days after sowing

**4 - SimStartMatu2** -INOUT- (en d)

procedure RS\_EvalSimStartMatu2(const NumPhase, ChangePhase, NbJas : Double; var SimStartMatu2 : Double);

begin

try

if (NumPhase = 6) and (ChangePhase = 1) then

begin

SimStartMatu2 := NbJas

end;

except

AfficheMessageErreur('RS\_EvalSimStartMatu2', URisocas);

end;

end;

**Module n°23 - RS\_EvalSimStartPSP**

This modules identifies the days after sowing when BVP ends and PSP starts

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - ChangePhase** -IN- : ce booléen permet de savoir si la journée courante est une journée de changement de phase (facilite l'initialisation)

**3 - NbJAS** -IN- (en d) : days after sowing

**4 - SimStartPSP** -INOUT- (en d)

procedure RS\_EvalSimStartPSP(const NumPhase, ChangePhase, NbJas : Double; var SimStartPSP : Double);

begin

try

if (NumPhase = 3) and (ChangePhase = 1) then

begin

SimStartPSP := NbJas

end;

except

AfficheMessageErreur('RS\_EvalSimStartPSP', URisocas);

end;

end;

**Module n°24 - RS\_EvalDegresJourCorVitMoy\_V2**

This module calculates the thermal (heat) units (state variable **DegresDuJour** ) received by the crop on day (i), on the basis of atmospheric min and max T and the cardinal temperatures TBase, TOpt1, TOpt2 and TLim (crop parameters). A corrected term **DegresDuJourCor**  is calculated by taking into account physiological drought, through the drought state variable “Cstr” and the crop parameter “DEVcstr”. The latter should be set to “0” if no slowing effect of drought on development is considered. At DEVcstr=1, there is a proportional effect of development rate (e.g., at cstr=0.5, all development processes take twice as long). Intermediate values give intermediate effects based on an exponential function that ensures that at any setting, development rate will be zero at cstr=0.

**1 - TMax** -IN- (en °C) : Température maximale mesurée

**2 - TMin** -IN- (en °C) : Température minimale mesurée

**3 - TBase** -IN- (en °C) : Base temperature (air based in this model; no microclimate simulated)

**4 - TOpt1** -IN- (en °C) : Lower limit of plateau of Thermal response of development

**5 - TOpt2** -IN- (en °C) : Upper limit of plateau of Thermal response of development

**6 - Tlim** -IN- (en °C) : Upper thermal limit of development

**7 - Cstr** -IN- (en none) : drought stress coefficient: FTSW is transformed into Cstr by FAO function using P-factor

**8 - DEVcstr** -IN- (en none) : Stress brake on development rate. 0=no effect, 1 = reduction in development rate is proportional to cstr. Intermediate levels are non-linear

**9 - StressCold** -IN- (en Coeff x)

**10 - DegresDuJour** -OUT- (en °C.d) : daily heat dose (in degree-days)

**11 - DegresDuJourCor** -OUT- (en °C.d) : same, but adjusted for drought effect using a value >0 for DEVcstr: drought slows development, thus reducing the effective heat dose available

procedure RS\_EvalDegresJourVitMoy\_V2(const NumPhase, TMax, TMin, TBase, TOpt1, TOpt2, TLet, cstr, DEVcstr, StressCold : double; var DegresDuJour, DegresDuJourCor : Double);

var

v, v1, v3 : Double;

S1, S2, S3 : Double;

Tn, Tx : Double;

begin

try

if (TMax <> TMin) then

begin

if ((TMax <= Tbase) or (TMin >= TLet)) then

begin

V := 0;

end

else

begin

Tn := Max(TMin, Tbase);

Tx := Min(TMax, TLet);

V1 := ((Tn + Min(TOpt1, Tx)) / 2 - Tbase) / (TOpt1 - Tbase);

S1 := V1 \* Max(0, min(TOpt1, Tx) - Tn) ;

S2 := 1 \* Max(0, min(Tx, TOpt2) - Max(Tn, TOpt1));

V3 := (TLet - (Max(Tx, TOpt2) + Max(TOpt2, Tn)) / 2) / (TLet - TOpt2);

S3 := V3 \* Max(0, Tx - Max(TOpt2, Tn));

V := (S1 + S2 + S3) / (TMax - TMin);

end

end

else

begin

if (TMax < TOpt1) then

begin

V := (TMax - Tbase) / (TOpt1 - Tbase);

end

else

begin

if (TMax < TOpt2) then

begin

V := 1

end

else

begin

V := (TLet - TMax) / (Tlet - TOpt2);

end;

end;

end;

DegresDuJour:= V \* (TOpt1 - TBase);

if (NumPhase > 1) and (NumPhase < 5) then

begin

DegresDuJourCor := DegresDuJour \* Power(Max(cstr, 0.00000001), DEVcstr);

end

else

begin

DegresDuJourCor := DegresDuJour;

end;

DegresDuJourCor := DegresDuJourCor \* StressCold;

except

AfficheMessageErreur('RS\_EvalDegresJourVitMoy | TMax='+FloatToStr(TMax)+

' TMin='+FloatToStr(TMin)+ 'TBase='+FloatToStr(TBase)+' TOpt1='+FloatToStr(TOpt1)+

' TOpt2='+FloatToStr(TOpt2)+' TL='+FloatToStr(TLet)+' DegresDuJour='+

FloatToStr(DegresDuJour)+' DegreDuJourCor='+FloatToStr(DegresDuJourCor),URisocas);

end;

end;

**Module n°25 - RS\_EvalSDJPhase4**

A specific counter needed to calculate progress within NumPhase 4 (reproductive). This is needed to define further down sub-phases of sensitivity of spikelet sterility to thermal and drought stresses.

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - DegresDuJourCor** -IN- (en °C.d) : same, but adjusted for drought effect using a value >0 for DEVcstr: drought slows development, thus reducing the effective heat dose available

**3 - SDJCorPhase4** -INOUT- (en °C.jour)

procedure RS\_EvalSDJPhase4(const NumPhase, DegreDuJour : Double; var SDJPhase4 : Double);

begin

try

if (NumPhase = 4) then

begin

SDJPhase4 := SDJPhase4 + DegreDuJour;

end;

except

AfficheMessageErreur('RS\_EvalSDJPhase4', URisocas);

end;

end;

**Module n°26 - RS\_EvalDAF\_V2**

A specific counter for time elapsing after flowering (DAF = days after flowering), needed to manage terminal drainage set by user under lowland conditions.

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - DAF** -INOUT- (en d)

procedure RS\_EvalDAF\_V2(const NumPhase : Double; var DAF : Double);

begin

try

if (NumPhase > 4) then

begin

DAF := DAF + 1;

end

else

begin

DAF := DAF;

end;

except

AfficheMessageErreur('RS\_EvalDAF\_V2', URisocas);

end;

end;

**Module n°27 - RS\_Phyllochron**

This module calculates the phyllochron (thermal time elapsing between two successive leaf appearances). It is an important process in SAMARA because it drives the demand for assimilates related to new organs on a phytomer, including leaf blades, sheaths and internodes. Since tillers are considered to have synchronized development (cohorts), they multiply this demand proportionally. Parameter “**Phyllo**” sets the basic (primary) phyllochron implemented during the vegetative growth stages (BVP & PSP), from the 4th leaf until onset of stem elongation. Stem elongation (set by binary state variable “PhaseStemElongation”) starts at panicle initiation (onset NumPhase 4 = reproductive phase), or on the 20th leaf, whatever happens first. [Clerget found that in sorghum, stem elongation starts on the 20th leaf if PI is late.] During stem elongation, phyllochron is longer, set by parameter **RelPhylloPhaseStemElong** (development slows down). Note: the first 3 leaves appear more rapidly than the others (phyllochron \* 0.5) because they are already pre-formed in the embryo, and need not be initiated any more. This is commonly observed in cereals.

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - DegresDuJourCor** -IN- (en °C.d) : same, but adjusted for drought effect using a value >0 for DEVcstr: drought slows development, thus reducing the effective heat dose available

**3 - Phyllo** -IN- (en °C.d) : Phyllochron (initial rate). Sets duration from one leaf appearance to the next. From internode elongation onwards phyllochron duration doubles

**4 - RelPhylloPhaseStemElong** -IN- : Sets degree of slow-down of development rate (1/phyllo) during stem elongation. Phyllochron doubles at value=0.5, remains constant at value=1

**5 - PhaseStemElongation** -OUT- (en none) : Indicates whether internodes are elongating (1) or not (0)

**6 - HaunGain** -OUT-

**7 - HaunIndex** -INOUT- (en none) : Number of leaves appeared on main stem, including those that have already senesced

procedure RS\_Phyllochron(const NumPhase, DegresDuJourCor, Phyllo, RelPhylloPhaseStemElong : Double; PhaseStemElongation, HaunGain, HaunIndex : Double);

begin

try

if ((NumPhase > 1) and (NumPhase < 5)) then

begin

if (((NumPhase > 3) or (HaunIndex > 20)) and (NumPhase < 5)) then

begin

PhaseStemElongation := 1;

end

else

begin

PhaseStemElongation := 0;

end;

if (PhaseStemElongation = 0) then

begin

HaunGain := DegresDuJourCor / Phyllo;

if (HaunIndex < 3) then

begin

HaunGain := HaunGain \* 2;

end;

end

else

begin

if (PhaseStemElongation = 1) then

begin

HaunGain := RelPhylloPhaseStemElong \* (DegresDuJourCor / Phyllo);

end;

end;

HaunIndex := HaunIndex + HaunGain;

end

else

begin

HaunGain := 0;

PhaseStemElongation := 0;

end;

except

AfficheMessageErreur('RS\_Phyllochron', URisocas);

end;

end;

**Module n°28 - RS\_EvolHauteur\_SDJ\_cstr**

This module calculates plant height, apex height and plant width. Plant height and width are essentially needed to simulate clumping effects on light interception. Apex height will be needed in order to simulate meristem temperature, particularly for flooded rice where floodwater temperature affects phenology and cold-induced sterility. All three variables will be needed to calculate microclimate (SAMARa V3). Variable PlantHeight is derived from the leaf blade+sheath length of the latest developed leaf (= parameter LeafLengthMax \* the rel. length of current leag position), with number of corrections: (1) correction for leaf angle using Kdf as indicator, (2) multiplication with the mean Ic (limited to max 1) to account for past supply restrictions, and (3) addition of sheath length which is a function of leaf length. ApexHeight is also added to accound for elongated internodes if any. PlantWidh is calculated similarly based on leaf length (but without sheath), IcMean and Kdf, with the additional provision that tillers (=CulmsPerHill-1) each add 10% to width. If PhaseStemElongation = 1, ApexHeight is calculated incrementally (ApexHeightGain) as increase in leaf (phytomer) number (=HaunGain), multiplied by the potential individual internode length (parameter InternodeLengthMax), the dought stress coefficient (cstr) and the square root of Ic (here set to max 1). Drought thus has a proportional effect on elongation, and resource limitation a milder one, with the principle of the most limiting factor applied.The result is then multiplied with the parameter CoeffInternodeNum because in most cases, not only currently developing phytomers elongate but also some older ones. The parameter thus provides the option to multiply the number of elongation internodes beyond the one currently producing a leaf.

**1 - PhaseStemElongation** -IN- (en none) : Indicates whether internodes are elongating (1) or not (0)

**2 - CoeffInternodeNum** -IN- (en none) : If value is 1, only the number of internodes corresponding to the phyllochrons between onset elongation and flowering will elongate

**3 - HaunGain** -IN-

**4 - Cstr** -IN- (en none) : drought stress coefficient: FTSW is transformed into Cstr by FAO function using P-factor

**5 - InternodeLengthMax** -IN- (en mm) : Maximal individual length of elongated internode (may not be attainted if constraints)

**6 - RelPotLeafLength** -IN- (en fraction) : Relative length of leaf blades currently developing, or the last one that developed, on a 0.1 scale. 1=potential relative length of longest leaf

**7 - LeafLengthMax** -IN- (en mm) : Maximal individual length of the longest leaf blade (may not be attainted if constraints)

**8 - CulmsPerHill** -IN-

**9 - IcMean** -IN- (en none) : Accued mean of Ic

**10 - Kdf** -IN- (en none) : Sets extinction of incoming diffuse solar radiation by crop canopy as function of LAI. Value 0.4 = very erect leaves, 1 = horizontal leaves

**11 - Ic** -IN- (en g/g) : state variable "index of competition" = daily assimilate supply/demand

**12 - WtRatioLeafSheath** -IN- (en fraction)

**13 - StressCold** -IN- (en Coeff x)

**14 - CstrMean** -IN- (en none)

**15 - ApexHeightGain** -OUT- (en mm)

**16 - ApexHeight** -INOUT- (en mm) : Height of growing point over ground (excluding the panicle and its peduncle)

**17 - PlantHeight** -OUT- (en mm) : Overall height of plant incuding top leaves, assuming vertical orientation

**18 - PlantWidth** -OUT- (en mm) : Approximate plant width

procedure RS\_EvolHauteur\_SDJ\_cstr(const PhaseStemElongation, CoeffInternodeNum, HaunGain, cstr, InternodeLengthMax, RelPotLeafLength, LeafLengthMax, CulmsPerHill, IcMean, Kdf, Ic, WtRatioLeafSheath, StressCold, CstrMean : Double; var ApexHeightGain, ApexHeight, PlantHeight, PlantWidth : Double);

var

CorrectedCstrMean : Double;

begin

try

if (PhaseStemElongation = 1) then

begin

ApexHeightGain := HaunGain \* Min(Power(Min(Ic, 1), 0.5), cstr) \* StressCold \* InternodeLengthMax;

ApexHeightGain := ApexHeightGain \* CoeffInternodeNum;

end

else

begin

ApexHeightGain := 0;

end;

ApexHeight := ApexHeight + ApexHeightGain;

if (CstrMean <= 0) then

begin

CorrectedCstrMean := 1;

end

else

begin

CorrectedCstrMean := CstrMean;

end;

PlantHeight := ApexHeight + (1.5 \* (1 - Kdf) \* RelPotLeafLength \* LeafLengthMax \* Sqrt(IcMean) \* CorrectedCstrMean \* (1 + 1 / WtRatioLeafSheath));

PlantWidth := Kdf \* 2 \* Sqrt(IcMean) \* RelPotLeafLength \* LeafLengthMax \* Min(1.4, (1 + 0.1 \* (CulmsPerHill - 1)));

except

AfficheMessageErreur('RS\_EvolHauteur\_SDJ\_cstr', URisocas);

end;

end;

**Module n°29 - RS\_EvolKcpKceBilhy**

This module divides the crop coefficient KcMax (which is a coefficient translating potential evapotranspiration (ETo or PET or ETP) into the maximal ET of the crop-soil system) into a soil surface and plant component, proportionally to the fraction of light hitting the soil (Kce) or the plant (kcp).

**1 - LTRkdfcl** -IN- (en fraction) : Light transmission rate of canopy as calculated with Kdfcl (taking into account crop Kdf and clumping), = 1-LIRkdfcl

**2 - KcMax** -IN- (en fraction) : FAO reference coefficient for crop canopy ET as fraction of PET

**3 - Mulch** -IN- (en %) : Coefficient de mulching (couvert paillis...) et/ou "auto-mulch" (rugosité du sol…), 1 pas d'effet mulch.

**4 - Kcp** -OUT- (en fraction) : Partial Kc (simulated current crop coefficient ETR/Eto) attributable to plant transpiration

**5 - Kce** -OUT- (en fraction) : Partial Kc (simulated current crop coefficient ETR/Eto) attributable to soil evaporation

**6 - KcTto** -OUT- (en fraction) : Total Kc (simulated current crop coefficient ETR/Eto) attributable to soil evaporation

procedure RS\_EvolKcpKceBilhy(const LTRkdfcl, KcMax, Mulch : Double; var Kcp, Kce, KcTot : Double);

begin

try

Kcp := Min((1 - LTRkdfcl) \* KcMax, KcMax);

Kcp := Min(Kcp, KcMax);

Kce := LTRkdfcl \* 1 \* (Mulch / 100);

KcTot := Kcp + Kce;

except

AfficheMessageErreur('RS\_BilhyEvolKcpLai', URisocas);

end;

end;

**Module n°30 - RS\_EvalEvapPot**

This module calculates potential soil surface evaporation bu mutiplying Kce with atmospheric demand (ETP).

**1 - ETo** -IN- (en mm/d) : potential evapotranspiration (FAO, also called PET, ETP or Eto). Approximates atmospheric demand for water vapor applied to a calm water surface

**2 - Kce** -IN- (en fraction) : Partial Kc (simulated current crop coefficient ETR/Eto) attributable to soil evaporation

**3 - EvapPot** -OUT- (en mm/d) : Potential soilsurface evaporation (taking into account effect of ground cover) assuming soil is saturated

procedure RS\_EvalEvapPot(const Etp, Kce : Double; var EvapPot : Double);

begin

try

EvapPot := Kce \* Etp;

except

AfficheMessageErreur('RS\_EvalEvapPot', URisocas);

end;

end;

**Module n°31 - RS\_EvolEvapSurfRFE\_RDE\_V2**

This module calculates soil surface evaporation (Evap) on the basis of topsoil (EpaisseurSurf), unless the system is bunded (BundHeight>0) and there is water in the Stockmacropores and/or Floodwater; and the water storage in the surface compartment (StoclSurface), root zone (StockRac) and total soil profile (StockTotal). The stock in the surface compartment is divided into a easily evaporable fraction (ValDFE) and a …

Kr is the calculated coefficient of reduction of potential soil surface evaporation due to water deficit.

(More follows…)

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - Kce** -IN- (en fraction) : Partial Kc (simulated current crop coefficient ETR/Eto) attributable to soil evaporation

**3 - EvapPot** -IN- (en mm/d) : Potential soilsurface evaporation (taking into account effect of ground cover) assuming soil is saturated

**4 - CapaREvap** -IN- (en mm) : Capacité du réservoir d'évaporation

**5 - CapaRDE** -IN- (en mm) : Réserve difficilement transpirable mais évaporable

**6 - CapaRFE** -IN- (en mm) : Capacité du réservoir facilement évaporable (au potentiel)

**7 - RuRac** -IN- (en mm) : Water column that can potentially be strored in soil volume explored by root system

**8 - RuSurf** -IN- (en mm) : Reserve utile de l'horizon de surface

**9 - FloodwaterDepth** -IN- (en mm)

**10 - BundHeight** -IN- (en mm) : Bunds leading to surface floodwater storage. No lateral seepage is simulated

**11 - EpaisseurSurf** -IN- (en mm) : Epaisseur de l'horizon de surface

**12 - EpaisseurProf** -IN- (en mm) : Epaisseur de l'horizon de profondeur

**13 - StockMacropores** -IN-

**14 - RootFront** -IN- (en mm) : depth of root front

**15 - ResUtil** -IN- (en mm/m)

**16 - Evap** -OUT- (en mm/d) : Actual soil surface evaporation under crop (if any is present)

**17 - ValRSurf** -INOUT- (en mm) : Contenu des 2 réservoirs RDE et REvap

**18 - ValRFE** -INOUT- (en mm) : Contenu de la RFE

**19 - ValRDE** -INOUT- (en mm) : Contenu de la RDE

**20 - StockRac** -INOUT- (en mm) : Water column stored in soil volume explored by root system

**21 - StockTotal** -INOUT- (en mm) : Total water column stored in soil profile

**22 - StockSurface** -INOUT- (en mm) : Water column stored in topsoil layer

**23 - Kr** -OUT- : Coefficient de réduction de l'évaporation potentielle

procedure RS\_EvolEvapSurfRFE\_RDE\_V2(const NumPhase, Kce, EvapPot, CapaREvap, CapaRDE, CapaRFE, RuRac, RuSurf, FloodwaterDepth, BundHeight, EpaisseurSurf, EpaisseurProf, StockMacropores, RootFront, RU : Double; var Evap, ValRSurf, ValRFE, ValRDE, StockRac, StockTotal, StockSurface, Kr, KceReal : Double);

var

ValRSurfPrec, EvapRU : Double;

Evap1, Evap2 : Double;

begin

try

if ((StockMacropores + FloodwaterDepth) = 0) or (NumPhase = 0) then

begin

ValRSurfPrec := ValRSurf;

// ValRSurf est l'eau contenue dans les réservoirs Revap (non transpirable) et RDE (transpirable et évaporable

if (ValRFE > 0) then

begin

if (ValRFE < EvapPot) then

begin

Evap1 := ValRFE;

Evap2 := Max(0, Min(ValRSurf, ((EvapPot - ValRFE) \* ValRSurf) / (CapaREvap + CapaRDE))); // borné à 0 et ValRSurf le 27/04/05

end

else

begin

Evap1 := EvapPot;

Evap2 := 0;

end;

end

else

begin

Evap1 := 0;

Evap2 := Max(0, Min(ValRSurf, EvapPot \* ValRSurf /( CapaREvap + CapaRDE))); // borné à 0 et ValRSurf le 27/04/05

end;

Evap := Evap1 + Evap2;

ValRFE := ValRFE - Evap1;

ValRSurf := ValRSurf - Evap2;

ValRDE := Max(0, ValRSurf - CapaREvap);

if (EvapPot = 0) then

begin

Kr := 0;

end

else

begin

Kr := Evap / EvapPot;

end;

// part de l'évaporation prélevée dans les réservoirs RFE et RDE

if (ValRSurf >= CapaREvap) then

begin

EvapRU := Evap;

end

else

begin

if (ValRSurfPrec <= CapaREvap) then

begin

EvapRU := Evap1;

end

else

begin

EvapRU := evap1 + ValRSurfPrec - CapaREvap;

end;

end;

//Evaporation de Ru et Rur, MAJ

if (RuRac <= RuSurf) then

begin

// quand les racines n'ont pas dépassé la première couche

StockRac := Max(0, StockRac - EvapRU \* RuRac / RuSurf);

end

else

begin

StockRac := Max(0, StockRac - EvapRU);

end;

StockTotal := StockTotal - EvapRU;

StockRac := Min(StockRac, StockTotal);

KceReal := Kce \* Kr;

end;

if (StockMacropores + FloodwaterDepth > 0) and (NumPhase > 0) then

begin

Evap := EvapPot;

ValRSurf := CapaREvap + StockMacropores \* (EpaisseurSurf / (EpaisseurSurf + EpaisseurProf));

ValRFE := CapaRFE + StockMacropores \* (EpaisseurSurf / (EpaisseurSurf + EpaisseurProf));

ValRDE := CapaRDE;

StockRac:= RuRac + StockMacropores \* (RootFront / (EpaisseurSurf + EpaisseurProf));

StockSurface:= RuSurf + StockMacropores \* (EpaisseurSurf / (EpaisseurSurf + EpaisseurProf));

StockTotal := (EpaisseurSurf + EpaisseurProf) \* RU / 1000 + StockMacropores;

StockRac := Min(StockRac, StockTotal);

Kr := 1;

KceReal := Kce;

end;

except

AfficheMessageErreur('RS\_EvolEvapSurfRFE\_RDE\_V2',URisocas);

end;

end;

**Module n°32 - RS\_EvalFTSW\_V2**

This module calculates the Fraction of Transpirable Soil Water (FTSW) as the ratio of plant-available water in the root zone (StockRac) over the potential transpirable water reserve in the same compartment (RuRac). RuRac does not include water in macropores that is potentially drainable (present under water logged conditions when plots are bunded and drainage (Dr) is limited by PercolationMax.). Under upland conditions (BundHeight=0), Stockrac is <= RuRac and FTSW is always <= 1. Under lowland conditions (BundHeight>0), StockRac can exceed Rurac and FTSW can be >1. FTSW is needed to calculate restrictions to transpiration (FAO P-Factor model) and to calculate drought induced spikelet sterility. It is not calculated (=0) when there is no plant (NumPhase = 0 or >6).

**1 - RuRac** -IN- (en mm) : Water column that can potentially be strored in soil volume explored by root system

**2 - StockTotal** -IN- (en mm) : Total water column stored in soil profile

**3 - StockMacropores** -IN-

**4 - StRuMax** -IN- (en mm) : Capacité maximale de la RU

**5 - StockRac** -INOUT- (en mm) : Water column stored in soil volume explored by root system

**6 - FTSW** -OUT- (en none) : fraction of transpirable soil water within the bulk root zone

procedure RS\_EvalFTSW\_V2(const RuRac, StockTotal, StockMacropores, StRuMax : Double; var StockRac, ftsw : Double);

begin

try

StockRac := Min(StockRac, (RuRac + (StockMacropores \* RuRac / StRuMax)));

StockRac := Min(StockRac, StockTotal);

if (RuRac > 0) then

begin

ftsw := StockRac / RuRac;

end

else

begin

ftsw := 0;

end;

except

AfficheMessageErreur('EvalFTSW | StRurMax: '+FloatToStr(RuRac)+' StRur: '+FloatToStr(StockRac)+' ftsw: '+FloatToStr(ftsw),URisocas);

end;

end;

**Module n°33 - RS\_EvalCstrPFactorFAO\_V2**

This module calculates Cstr, the plant stress coefficient governing transpiration under drought. It uses FTSW and transforms it according to a broken-stick function using the FAO P-Factor (crop parameter PFactor), which idefines how much FTSW has to decrease below 1 until stomata begin to close. From that point onwards, transpiration linearly decreases and attains 0 at FTSW=0. Cstr is needed to calculate a number of drought stress responses, namely Tr=Cstr\*TrPot.

Attention: the stress coefficient cstr also takes care of water logging effects because in sensitive plants, this stress causes stomatal closure just like its opposite, drought.

**1 - PFactor** -IN- (en none) : FAO reference for critical FTSW value for transpiration response. Value 0 = stomata respond immediately if FTSW<1. Most crops are around 0.5

**2 - FTSW** -IN- (en none) : fraction of transpirable soil water within the bulk root zone

**3 - ETo** -IN- (en mm/d) : potential evapotranspiration (FAO, also called PET, ETP or Eto). Approximates atmospheric demand for water vapor applied to a calm water surface

**4 - KcTot** -IN- (en fraction) :

**5 - StockMacropores** -IN-

**6 - CoeffStressLogging** -IN- (en none)

**7 - Cstr** -OUT- (en none) : drought stress coefficient: FTSW is transformed into Cstr by FAO function using P-factor

procedure RS\_EvalCstrPFactorFAO\_V2(const PFactor, FTSW, ETo, KcTot, StockMacropores, CoeffStressLogging : Double; var cstr : Double);

var

pFact : Extended;

begin

try

pFact := PFactor + 0.04 \* (5 - KcTot \* ETo);

pFact := Max(0, pFact);

pFact := Min(0.8, pFact);

cstr := Min((FTSW / (1 - pFact)), 1);

cstr := Max(0, cstr);

if (StockMacropores > 0) then

begin

cstr := cstr \* CoeffStressLogging;

end;

except

AfficheMessageErreur('RS\_EvalCstrPFactorFAO\_V2',URisocas);

end;

end;

**Module n°34 - BhyCropWaterNeed**

This module calculates potential transpiration (TrPot) by multiplying evaporative demand (ETo) with the plant coefficient Kcp. Kcp is fraction of the crop coefficient Kcmax attributed to the soil surface covered by plants.

**1 - Kcp** -IN- (en fraction) : Partial Kc (simulated current crop coefficient ETR/Eto) attributable to plant transpiration

**2 - ETo** -IN- (en mm/d) : potential evapotranspiration (FAO, also called PET, ETP or Eto). Approximates atmospheric demand for water vapor applied to a calm water surface

**3 - TrPot** -OUT- (en mm/d) : Potential crop transpiration taking into account LAI and drought level (cstr)

procedure DemandePlante(const Kcp, ETo : Double; var TrPot : Double);

begin

try

TrPot := Kcp \* ETo;

except

AfficheMessageErreur('DemandePlante',UBilEau);

end;

end;

**Module n°35 - BhyTranspi**

This module calculates actual transpiration (Tr) by multiplying potential transpiration (TrPot) with the stress coefficient Cstr.

**1 - TrPot** -IN- (en mm/d) : Potential crop transpiration taking into account LAI and drought level (cstr)

**2 - Cstr** -IN- (en none) : drought stress coefficient: FTSW is transformed into Cstr by FAO function using P-factor

**3 - Tr** -OUT- (en mm/d) : Actual crop transpiration

procedure EvalTranspi(const TrPot, cstr : Double; var Tr : Double);

begin

try

Tr := TrPot \* cstr;

except

AfficheMessageErreur('EvalTranspi',UBilEau);

end;

end;

**Module n°36 - BilhyETRETM**

This module calculates output variables ETM (maximal evapotranspiration in the absence of water deficit including soil and plan surface) and ETR (real evapotranspiration in the presence of water deficit including soil and plan surface).

**1 - Evap** -IN- (en mm/d) : Actual soil surface evaporation under crop (if any is present)

**2 - Tr** -IN- (en mm/d) : Actual crop transpiration

**3 - TrPot** -IN- (en mm/d) : Potential crop transpiration taking into account LAI and drought level (cstr)

**4 - ETM** -OUT- (en mm/d) : Maximal ET of crop taking into accoung crop Kc and current LAI

**5 - ETR** -OUT- (en mm/d) : Actual ET of crop taking into account crop Kc, current LAI and Cstr (causing drought induced stomatal clusure)

procedure EvalETRETM (const Evap, Tr, Trpot : Double; var ETM, ETR : Double);

begin

try

ETM:= Evap+Trpot;

ETR:= Evap+Tr;

except

AfficheMessageErreur('EvalETRETM',UBhyTypeFAO);

end;

end;

**Module n°37 - RS\_EvolConsRes\_Flood\_V2**

This module recalculates soil water relations after the extraction of water consumption by soil evaporation (Evap) and plant transpiration (Tr). The routine used in SARRAH is applied if there is no water logging (water in macropores and/or floodwater under bunded condition). This calculation treats the soil surface and deep compartments separately. If there is water logging, Evap and Tr are drawn from floodwater and StockMacropores first, and only the remainder from the available soil water stock (micropores).

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - RuRac** -IN- (en mm) : Water column that can potentially be strored in soil volume explored by root system

**3 - RuSurf** -IN- (en mm) : Reserve utile de l'horizon de surface

**4 - CapaREvap** -IN- (en mm) : Capacité du réservoir d'évaporation

**5 - Tr** -IN- (en mm/d) : Actual crop transpiration

**6 - Evap** -IN- (en mm/d) : Actual soil surface evaporation under crop (if any is present)

**7 - CapaRDE** -IN- (en mm) : Réserve difficilement transpirable mais évaporable

**8 - CapaRFE** -IN- (en mm) : Capacité du réservoir facilement évaporable (au potentiel)

**9 - EpaisseurSurf** -IN- (en mm) : Epaisseur de l'horizon de surface

**10 - EpaisseurProf** -IN- (en mm) : Epaisseur de l'horizon de profondeur

**11 - ResUtil** -IN- (en mm/m)

**12 - StockRac** -INOUT- (en mm) : Water column stored in soil volume explored by root system

**13 - StockSurface** -INOUT- (en mm) : Water column stored in topsoil layer

**14 - StockTotal** -INOUT- (en mm) : Total water column stored in soil profile

**15 - ValRFE** -INOUT- (en mm) : Contenu de la RFE

**16 - ValRDE** -INOUT- (en mm) : Contenu de la RDE

**17 - ValRSurf** -INOUT- (en mm) : Contenu des 2 réservoirs RDE et REvap

**18 - FloodwaterDepth** -INOUT- (en mm)

**19 - StockMacropores** -INOUT-

procedure RS\_EvolConsRes\_Flood\_V2(const NumPhase, RuRac, RuSurf, CapaREvap, Tr, Evap, CapaRDE, CapaRFE, EpaisseurSurf, EpaisseurProf, Ru : Double; var StockRac, StockSurface, StockTotal, ValRFE, ValRDE, ValRSurf, FloodwaterDepth, StockMacropores : Double);

var

TrSurf : Double;

WaterDeficit : Double;

begin

try

TrSurf := 0;

if (FloodwaterDepth + StockMacropores = 0) or (NumPhase = 0) then

begin

// le calcul de cstr et de Tr doit intervenir après l'évaporation

// calcul de la part de transpiration affectée aux réservoirs de surface

if (RuRac <> 0) then

begin

if (RuRac <= RuSurf)

begin

TrSurf:=Tr;

end

else

begin

if (StockRac <> 0) then

begin

TrSurf := Tr \* StockSurface / StockRac; // pondération par les stocks et non les capacités, sinon on n'extrait pas Tr si stock nul

end;

end;

end

else

begin

TrSurf := 0;

end;

// MAJ des réservoirs de surface en répartissant TrSurf entre RFE et RDE

ValRDE := Max(0, ValRSurf - CapaREvap);

if (ValRDE + ValRFE < TrSurf) then

begin

ValRFE := 0;

ValRSurf := ValRSurf - ValRDE;

ValRDE := 0;

end

else

begin

if (ValRFE > TrSurf) then

begin

ValRFE := ValRFE - TrSurf; // priorité à la RFU

end

else

begin

ValRSurf := ValRSurf - (TrSurf - ValRFE);

ValRDE := ValRDE - (TrSurf - ValRFE);

ValRFE := 0;

end;

end;

StockSurface := ValRFE + ValRDE;

StockRac := Max(0, StockRac – Tr

StockTotal := Max(0, StockTotal - Tr);

StockRac := Min(StockRac, StockTotal);

end;

if ((StockMacropores + FloodwaterDepth) > 0) and ((StockMacropores + FloodwaterDepth) <= (Tr + Evap)) and (NumPhase > 0) then

begin

WaterDeficit := (Tr + Evap) - (StockMacropores + FloodwaterDepth);

StockMacropores := 0;

FloodwaterDepth := 0;

StockTotal := (EpaisseurSurf + EpaisseurProf) \* RU / 1000 - WaterDeficit;

StockRac := RuRac - WaterDeficit;

StockRac := Min(StockRac, StockTotal);

StockSurface := Max(EpaisseurSurf \* RU / 1000 - WaterDeficit, 0);

ValRFE := Max(StockSurface - ValRDE - Waterdeficit, 0);

ValRDE := ValRDE;

ValRSurf := ValRFE + ValRDE;

end

else

begin

if ((StockMacropores + FloodwaterDepth) > (Tr + Evap)) and (NumPhase > 0) then

begin

FloodwaterDepth := FloodwaterDepth - (Tr + Evap);

StockMacropores := StockMacropores + Min(0, FloodwaterDepth);

FloodwaterDepth := Max(FloodwaterDepth, 0);

StockTotal := (EpaisseurSurf + EpaisseurProf) \* RU / 1000 + StockMacropores;

StockRac := RuRac + StockMacropores;

StockRac := Min(StockRac, StockTotal);

StockSurface := Max(EpaisseurSurf \* RU / 1000 + StockMacropores \* (EpaisseurSurf / (EpaisseurSurf + EpaisseurProf)), 0);

ValRFE := Max(StockSurface - ValRDE, 0);

ValRDE := ValRDE;

end;

end;

except

AfficheMessageErreur('RS\_EvolConsRes\_Flood\_V2',URisocas);

end;

end;

**Module n°38 - RS\_EvalTMaxMoy**

This module calculates the thermal conditions during the sub-phase sensitive to heat induced spikelet sterility (just before and at flowering).

**1 - TMax** -IN- (en °C) : Température maximale mesurée

**2 - NumPhase** -IN- (en none) : Phenological phase

**3 - NumSsPhase** -IN-

**4 - TmaxMoy** -INOUT- (en °C) : Mean Tmax observed during critical period for heat induced spikelet sterility

procedure RS\_EvalTMaxMoy(const TMax, NumPhase, NumSousPhase : Double; var TMaxMoy : double) ;

begin

try

if ((NumPhase = 4) and (NumSousPhase = 4)) then

CalculeLaMoyenne(TMax, MonCompteur, TMaxMoy)

else

if NumPhase < 4 then

TMaxMoy:=0;

except

AfficheMessageErreur('RS\_EvalTMaxMoy',URiz);

end;

end;

**Module n°39 - RS\_EvalTMinMoy**

This module calculates the thermal conditions during the sub-phase sensitive to cold induced spikelet sterility (2 weeks to 1 week before flowering, roughly microspore stage).

**1 - TMin** -IN- (en °C) : Température minimale mesurée

**2 - NumPhase** -IN- (en none) : Phenological phase

**3 - NumSsPhase** -IN-

**4 - TminMoy** -INOUT- (en °C) : Mean Tmin observed during critical period for cold induced spikelet sterility

procedure RS\_EvalTMinMoy(const TMin, NumPhase, NumSousPhase : Double; var TMinMoy : double);

begin

try

if ((NumPhase = 4) and (NumSousPhase = 3)) then

begin

CalculeLaMoyenne(TMin, MonCompteur, TMinMoy);

end

else

begin

if NumPhase < 4 then

begin

TMinMoy := 0;

end;

end;

except

AfficheMessageErreur('RS\_EvalTMinMoy',URiz);

end;

end;

**Module n°40 - RS\_EvalFtswMoy**

This module calculates the mean FTSW during the sub-phase sensitive to drought induced spikelet sterility (just before and at flowering).

**1 - FTSW** -IN- (en none) : fraction of transpirable soil water within the bulk root zone

**2 - NumPhase** -IN- (en none) : Phenological phase

**3 - NumSsPhase** -IN-

**4 - FtswMoy** -INOUT- (en fraction) : Mean FTSW observed during critical period for drought induced spikelet sterility

procedure RS\_EvalFtswMoy(const Ftsw, NumPhase, NumSousPhase : Double; var FtswMoy : double);

begin

try

if ((NumPhase = 4) and (NumSousPhase = 4)) then

begin

CalculeLaMoyenne(Ftsw, MonCompteur, FtswMoy);

end

else

begin

if NumPhase < 4 then

begin

FtswMoy := 0;

end;

end;

except

AfficheMessageErreur('RS\_EvalFtswMoy',URiz);

end;

end;

**Module n°41 - RS\_EvalSterility**

This module calcules cold-, heat- and drought induced spikelet sterility, as well as total sterility (as a fraction of total spikelet number). For each component of sterility, two crop parameters are used (Kcrit…1 and Kcrit…2), the first representing the conditions under which sterility begins to occur, and the second conditions where sterility is total. Note that total sterility is not the simple sum of sterility components because it cannot be >1!

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - ChangePhase** -IN- : ce booléen permet de savoir si la journée courante est une journée de changement de phase (facilite l'initialisation)

**3 - KCritSterCold1** -IN- (en °C) : Daily min temperature at pre-flowering below which there may be cold-induced sterility

**4 - KCritSterCold2** -IN- (en °C) : Daily min temperature at which cold-induced sterility attains 100%

**5 - KCritSterHeat1** -IN- (en °C) : Daily Max temperature around flowering above which heat induces sterility

**6 - KCritSterHeat2** -IN- (en °C) : Daily Max temperature around flowering above which heat induced sterility is 100%

**7 - KCritSterFtsw1** -IN- (en fraction) : FTSW value around flowering below which drought induced sterility is observed

**8 - KCritSterFtsw2** -IN- (en fraction) : FTSW value around flowering below which drought induced sterility is 100%

**9 - TminMoy** -IN- (en °C) : Mean Tmin observed during critical period for cold induced spikelet sterility

**10 - TmaxMoy** -IN- (en °C) : Mean Tmax observed during critical period for heat induced spikelet sterility

**11 - FtswMoy** -IN- (en fraction) : Mean FTSW observed during critical period for drought induced spikelet sterility

**12 - SterilityCold** -INOUT- (en fraction) : Spikelet sterility due to low temperatures during microspore stage (ca booting stage) based on daily Tmin during sensitive period

**13 - SterilityHeat** -INOUT- (en fraction) : Spikelet sterility due to high temperatures during heading/flowering stage based on daily Tmax during sensitive period

**14 - SterilityDrought** -INOUT- (en fraction) : Spikelet sterility due to frought (as indicated by FTSW) during heading/flowering stage

**15 - SterilityTot** -INOUT- (en fraction) : Total spikelet sterility (caused by cold, heat and drought)

procedure RS\_EvalSterility(const Numphase, ChangePhase, KCritSterCold1, KCritSterCold2, KCritSterHeat1, KCritSterHeat2, KCritSterFtsw1, KCritSterFtsw2, TMinMoy, TMaxMoy, FtswMoy : Double; var SterilityCold, SterilityHeat, SterilityDrought, SterilityTot : Double);

begin

try

if ((NumPhase = 5) and (ChangePhase = 1)) then

begin

SterilityCold := Max(0, (Min(1, KCritSterCold1 / (KCritSterCold1 - KCritSterCold2) - TMinMoy / (KCritSterCold1 - KCritSterCold2))));

SterilityHeat := 1 - Max(0, (Min(1, KCritSterHeat2 / (KCritSterHeat2 - KCritSterHeat1) - TMaxMoy / (KCritSterHeat2 - KCritSterHeat1))));

SterilityDrought := Max(0, (Min(1, KCritSterFtsw1 / (KCritSterFtsw1 - KCritSterFtsw2) - FtswMoy / (KCritSterFtsw1 - KCritSterFtsw2))));

end

else

begin

SterilityCold := SterilityCold;

SterilityHeat := SterilityHeat;

SterilityDrought := SterilityDrought;

end;

SterilityTot := Min(0.999, 1 - ((1 - SterilityCold) \* (1 - SterilityHeat) \* (1 - SterilityDrought)));

except

AfficheMessageErreur('RS\_EvalSterility', URisocas);

end;

end;

**Module n°42 - RS\_EvalVitesseRacinaire**

Recoding of maximal root front speed for the different growth phases. The parameter RootCstr (0…1) permits to optionally let Cstr impact on root growth, with value 0 for no effect, value 1 for proportional effect (inhibition) and intermediate values for intermediate effect.

**1 - VRacLevee** -IN- (en mm/d) : Root front advance per day in mm, provided the wetting front or pre-set soil depth doesn't stop it

**2 - VRacBVP** -IN- (en mm/d) : same for BVP

**3 - VRacRPR** -IN- (en mm/d) : same for reproductive phase

**4 - VRacPSP** -IN- (en mm/d) : same for PSP

**5 - VRacMatu1** -IN- (en mm/d) : same for grain filling phase

**6 - VRacMatu2** -IN- (en mm/d) : same for terminal mauration phase

**7 - RootCstr** -IN- (en none) : Attenuator of root front advancement as function of cstr (drought). No effect at value 0, proportional effect at value 1

**8 - Cstr** -IN- (en none) : drought stress coefficient: FTSW is transformed into Cstr by FAO function using P-factor

**9 - NumPhase** -IN- (en none) : Phenological phase

**10 - DegresDuJourCor** -IN- (en °C.d) : same, but adjusted for drought effect using a value >0 for DEVcstr: drought slows development, thus reducing the effective heat dose available

**11 - VitesseRacinaire** -OUT- (en mm/jour) : Vitesse racinaire journalière

**12 - VitesseRacinaireDay** -OUT- (en mm/d) : current progression rate of root front

procedure RS\_EvalVitesseRacinaire(const VRacLevee, RootSpeedBVP, RootSpeedRPR, RootSpeedPSP, RootSpeedMatu1, RootSpeedMatu2, RootCstr, cstr, NumPhase, DegreDuJour : Double; var VitesseRacinaire, VitesseRacinaireDay : Double);

begin

try

case Trunc(NumPhase) of

1 : VitesseRacinaire := VRacLevee;

2 : VitesseRacinaire := RootSpeedBVP;

3 : VitesseRacinaire := RootSpeedPSP;

4 : VitesseRacinaire := RootSpeedRPR;

5 : VitesseRacinaire := RootSpeedMatu1;

6 : VitesseRacinaire := RootSpeedMatu2;

else

VitesseRacinaire := 0

end;

VitesseRacinaireDay := VitesseRacinaire \* DegreDuJour \* Power(cstr, RootCstr);

except

AfficheMessageErreur('EvalVitesseRacinaire | NumPhase: ' + FloatToStr(NumPhase), URisocas);

end;

end;

**Module n°43 - EvalConversion**

This module implements the optional, NumPhase-specific modifiers of Epsib (called TxConversion in list! = potential radiation use efficiency). They are called AssimBVP, KAssimMati2… and should be used with caution, and never to make simulations fit to funny data, because this is strictly speaking a manipulation. The such modified Epsib coefficient is called Conversion.

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - TxConversion** -IN- (en g/MJ) : Potential radiation use efficiency (RUE=epsilon-b) BEFORE maintenance. This value can be up to 2x higher that RUE found in literature

**3 - TxAssimBVP** -IN- (en fraction) : Reduction factor to force lower assimilation during this phase

**4 - SumDegresDay** -IN- (en °C.jour) : Somme de degrés.jours depuis le début de la phase 1

**5 - SumDDPhasePrec** -IN- (en °C.jour) : Somme en degrés/jour de la phase précédente

**6 - TxAssimMatu1** -IN- (en fraction) : Reduction factor to force lower assimilation during this phase

**7 - TxAssimMatu2** -IN- (en fraction) : Reduction factor to force lower assimilation during this phase

**8 - SeuilTemp** -IN- (en °C.jour) : Seuil des températures cumulées pour la phase en cours

**9 - Conversion** -OUT- (en g/m²/MJ)

procedure EvalConversion(const NumPhase, EpsiB, AssimBVP, SommeDegresJour, SommeDegresJourPhasePrecedente, AssimMatu1, AssimMatu2, SeuilTempPhaseSuivante : Double; var Conversion : Double);

var

KAssim : Double;

begin

try

case Trunc(NumPhase) of

2 : KAssim := 1;

3..4 : KAssim := AssimBVP;

5 : KAssim := AssimBVP + (SommeDegresJour - SommeDegresJourPhasePrecedente) \*

(AssimMatu1 - AssimBVP) / (SeuilTempPhaseSuivante - SommeDegresJourPhasePrecedente);

6 : KAssim := AssimMatu1 + (SommeDegresJour - SommeDegresJourPhasePrecedente) \*

(AssimMatu2 - AssimMatu1) / (SeuilTempPhaseSuivante - SommeDegresJourPhasePrecedente);

else

KAssim := 0;

end;

Conversion:=KAssim\*EpsiB;

except

AfficheMessageErreur('EvalConversion | NumPhase: '+FloatToStr(NumPhase)+

' SommeDegresJour: '+FloatToStr(SommeDegresJour),UMilBilanCarbone);

end;

end;

**Module n°44 - RS\_EvalParIntercepte**

This module calculates intercepted PAR from incident PAR by multiplying it with (1-LTRkdfcl) . LTRkdfcl is the light transmission ratio based on an extinction coefficient for diffusive radiation Kdf modified by a clumping coefficient.

**1 - Par** -IN- (en MJ/m²/d) : Photosynthetically active radiation (PAR), which is about 50% of incoming global solar radiation

**2 - LIRkdfcl** -IN- (en fraction) : Light interception rate of canopy as calculated with Kdfcl (taking into account crop Kdf and clumping)

**3 - PARIntercepte** -OUT- (en MJ/m²/d) : PAR intercepted by crop

procedure RS\_EvalParIntercepte(Const PAR, LIRkdfcl : Double; var PARIntercepte : Double);

begin

try

PARIntercepte := PAR \* LIRkdfcl;

except

AfficheMessageErreur('RS\_EvalParIntercepte | PAR: '+ FloatToStr(PAR) + ' LIRkdfcl: ' + FloatToStr(LIRkdfcl), URisocas);

end;

end;

**Module n°45 - RS\_EvalAssimPot**

This module calculates potential canopy level assimilation (AssimPot, kg/ha/d) by multiplying intercepted PAR with Conversion. These are all state variables. The fixed coefficient of 10 takes car of unit conversion (PAR is based on /m²/d, Conversion on g/m²/d, AssimPot on kg/ha/d). The max function involving Tmax, Tmin, Tbase and Top1 takes care of a reduction in AssimPot if ambient T decreases below Topt1, AssimPot is zero at T=Tmin. The calculation of ambient T gives 3x greater weight to Tmax than to Tmin because photosynthesis happens only at day time. It must thus be noted that the genotypic choice of Tbase and Topt1 not only affects phenology but also photosynthesis, with a linear decrease from 100% at Topt1 to 0% at Tbase.

Attention: Because we found that the proportionality of AssimPot vs. PAR (linear response at canopy scale) leads to excessive sensitivity of growth to low light, we introduced a slightly convexe curvature of the reponse. This leaves AssimPot unaltered at full sunlit days (calculated from daily PAR normalized by daylength), but increases it at lower light (frequent) and increases it at higher light levels (which is exceptional). This is an empirical departure from the RUE=constant paradigm. Anyway, we already departed from it by implementing Rm independent of PAR.

**1 - PARIntercepte** -IN- (en MJ/m²/d) : PAR intercepted by crop

**2 - Conversion** -IN- (en g/m²/MJ)

**3 - TMax** -IN- (en °C) : Température maximale mesurée

**4 - TMin** -IN- (en °C) : Température minimale mesurée

**5 - TBase** -IN- (en °C) : Base temperature (air based in this model; no microclimate simulated)

**6 - TOpt1** -IN- (en °C) : Lower limit of plateau of Thermal response of development

**7 – DayLength** -IN- (en hour(dec)) : day length including civil twilight

**8 - AssimPot** -OUT- (en kg/ha/d) : Canopu CH20 assimilation per day BEFORE reduction by stomatal closure (mediated by Cstr) and subtraction of Rm

procedure RS\_EvalAssimPot(const PARIntercepte, Conversion, Tmax, Tmin, Tbase, Topt1, DayLength, StressCold : Double; var AssimPot : Double);

begin

try

AssimPot := PARIntercepte \* Conversion \* 10;

// Ordinary linear effect on intercepted light on canopy assimulation

AssimPot := AssimPot \* Min(((3 \* Tmax + Tmin) / 4 - Tbase) / (Topt1 - Tbase), 1);

// Reduction of assimilation at diurnal temperatures < Topt1

AssimPot := AssimPot \* Sqrt(Max(0.00001, StressCold));

// Cold stress effect on AssimPot (affects also organ demands and grain filling)

if ((PARIntercepte <> 0) and (DayLength <> 0)) then

begin

AssimPot := AssimPot \* Power(PARIntercepte / DayLength, 0.667) / (PARIntercepte / DayLength);

// De-linearization of PAR response of AssimPot. At 1 MJ/h (cloudless) effect is zero

end;

except

AfficheMessageErreur('RS\_EvalAssimPot', URisocas);

end;

end;

**Module n°46 - RS\_EvalCstrAssim**

This module calculates a coefficient (CstrAssim) that optionally modifies the proportionality between transpiration and assimilation decreases under drought (drought = situations of Cstr < 1). Proportionality is assumed if ASScstr=0. But this is unphysiological because CO2 exchange of leaves is less sensitive to stomatal closure than transpiration (which is why partial stomatal closure increases transpiration efficiency TE!). A value of 0.5 foe ASScstr is roughly appropriate, resulting in a curvilinear decline of AssimPot as relative transpiration (TR/TM = cstr) decreases linearly. Exact values still need to be determined for C3 and C4 plants separately.

**1 - Cstr** -IN- (en none) : drought stress coefficient: FTSW is transformed into Cstr by FAO function using P-factor

**2 - ASScstr** -IN- (en none) : Attenuator of A as a function of cstr (simulating drought effect on T)

**3 - CstrAssim** -OUT- (en Coeff x) : coeff de réduction de AssimPot en fonction de FTSW

procedure RS\_EvalCstrAssim(const cstr, ASScstr : Double; var cstrassim : Double);

begin

try

cstrassim := Power(Max(cstr, 0.00000001), ASScstr);

except

AfficheMessageErreur('RS\_EvalCstrAssim', URisocas);

end;

end;

**Module n°47 - RS\_EvalAssim**

This module calculates actual assimilation rate (Assim) by multiplying AssimPot with CstrAssim.

**1 - AssimPot** -IN- (en kg/ha/d) : Canopu CH20 assimilation per day BEFORE reduction by stomatal closure (mediated by Cstr) and subtraction of Rm

**2 - CstrAssim** -IN- (en Coeff x) : coeff de réduction de AssimPot en fonction de FTSW

**3 - Assim** -OUT- (en kg/ha/d) : Assim=AssimPot \* Cstr (if applicable, corrected with CstrAssim)

procedure RS\_EvalAssim(const AssimPot, CstrAssim : Double; var Assim : Double);

begin

try

Assim := Max(AssimPot \* CstrAssim, 0);

except

AfficheMessageErreur('EvalAssim | AssimPot: '+FloatToStr(AssimPot)+

' CstrAssim: '+FloatToStr(CstrAssim)+' StressCold: ', URisocas);

end;

end;

**Module n°48 - RS\_TransplantingShock\_V2**

Module calculating a decrease of photosynthesis (Assim) during the 1st 7 days after transplanting if parameter CoeffTransplantingShock < 1.

**1 - CounterNursery** -IN-

**2 - CoeffTransplantingShock** -IN- (en fraction)

**3 - Assim** -INOUT- (en kg/ha/d) : Assim=AssimPot \* Cstr (if applicable, corrected with CstrAssim)

procedure RS\_TransplantingShock\_V2(const CounterNursery, CoeffTransplantingShock : Double; var Assim : Double);

begin

try

if ((CounterNursery > 0) and (CounterNursery <8)) then

begin

Assim := Assim \* CoeffTransplantingShock;

end

else

begin

Assim := Assim;

end;

except

AfficheMessageErreur('RS\_TransplantingShock\_V2', URisocas);

end;

end;

**Module n°49 - RS\_EvalRespMaint**

Module calculating maintenance respiration (RespMaintTot) as the sum of RM of each organ class ; by multiplying organ structural dry matter with an organ specific respiration coefficient and CoeffQ10. CoeffQ10 is calculated from the crop parameter CoefficientQ10 and daily mean temperature using the Q10 rule, according to which the rate of the process increases by factor coefficientQ10 as T increases by 10 °C. The conventional value is Q10=2, but recent research indicated that under field conditions and with acclimation, Q10=1.5 is more accurate. The question remains open and is extremely relevant for climate change impact research.

**1 - kRespMaintLeaf** -IN- (en g/g) : Daily dw loss to Rm at reference temperture 25°C (fraction of current dw). For the organ concerned

**2 - kRespMaintSheath** -IN- (en g/g) : Daily dw loss to Rm at reference temperture 25°C (fraction of current dw). For the organ concerned

**3 - kRespMaintRoot** -IN- (en g/g) : Daily dw loss to Rm at reference temperture 25°C (fraction of current dw). For the organ concerned

**4 - kRespInternode** -IN- (en g/g) : Daily dw loss to Rm at reference temperture 25°C (fraction of current dw). For the organ concerned

**5 - kRespPanicle** -IN- (en g/g) : Daily dw loss to Rm at reference temperture 25°C (fraction of current dw). For the organ concerned

**6 - DryMatStructLeafPop** -IN- (en kg/ha) : Green leaf blade dry matter at population scale

**7 - DryMatStructSheathPop** -IN- (en kg/ha) : Sheath blade dry matter at population scale

**8 - DryMatStructRootPop** -IN- (en kg/ha) : Root blade dry matter at population scale

**9 - DryMatStructInternodePop** -IN- (en kg/ha) : Internode blade dry matter at population scale (only structural component: reserves are simulated and output separately)

structural component: reserves are simulated and output separately)

**10 - DryMatStructPaniclePop** -IN- (en kg/ha) : Panicle structural dry matter at population scale (does not include grains), formed between PI and flowering

include grains), formed between PI and flowering

**11 - TMoyCalc** -IN- (en °C) : Mean of Tmin and Tmax

**12 - kTempMaint** -IN- (en °C) : Température de référence de respiration de maintenance

**13 - CoefficientQ10** -IN- (en none) : Coefficient for Q10 rule for Rm. No effect at value 1, literature value of 2 doubles rate as T increases by 10°

**14 - RespMaintTot** -OUT- (en kg/ha/d) : Total daily maintenance respiration (Rm), sum of that of all organs as calculated with organ specific coefficients

procedure RS\_EvalRespMaint(const kRespMaintLeaf, kRespMaintSheath, kRespMaintRoot, kRespInternode, kRespPanicle : Double; const DryMatStructLeafPop, DryMatStructSheathPop, DryMatStructRootPop, DryMatStructInternodePop, DryMatStructPaniclePop : Double; const TMoyCalc, kTempMaint, CoefficientQ10 : Double; var RespMaintTot : Double);

var

RespMaintLeafPop : Double;

RespMaintSheathPop : Double;

RespMaintRootPop : Double;

RespMaintInternodePop : Double;

RespMaintPaniclePop : Double;

CoeffQ10 : Double;

begin

try

CoeffQ10 := Power(CoefficientQ10, (TMoyCalc - kTempMaint) / 10);

RespMaintLeafPop := kRespMaintLeaf \* DryMatStructLeafPop \* CoeffQ10;

RespMaintSheathPop := kRespMaintSheath \* DryMatStructSheathPop \* CoeffQ10;

RespMaintRootPop := kRespMaintRoot \* DryMatStructRootPop \* CoeffQ10;

RespMaintInternodePop := kRespInternode \* DryMatStructInternodePop \* CoeffQ10;

RespMaintPaniclePop := kRespPanicle \* DryMatStructPaniclePop \* CoeffQ10;

RespMaintTot := RespMaintLeafPop + RespMaintSheathPop + RespMaintRootPop + RespMaintInternodePop + RespMaintPaniclePop;

except

AfficheMessageErreur('RS\_EvalRespMaint', URisocas);

end;

end;

**Module n°50 - RS\_EvalRelPotLeafLength**

This module calculates the relative potential leaf length according to its rank on the main stem (state variable HaunIndex). It is assumed that the 1st leaf has 10% of the length of the longest leaf, and that leaf length on successive ranks increases linearly until the longest leaf is produced, by definition on rank RankLongestLeaf (crop parameter). Thereafter, potential leaf length remains constant. (The common observation that the flag leaf is shorter than its precursor is disregarded here for simplicity). RelPotLefLength is a relative, unitless value between 0.1 and 1.

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - HaunIndex** -IN- (en none) : Number of leaves appeared on main stem, including those that have already senesced

**3 - RankLongestLeaf** -IN- (en none) : Position of longest leaf on main stem, ususally between 10th and 15th

**4 - RelPotLeafLength** -OUT- (en fraction) : Relative length of leaf blades currently developing, or the last one that developed, on a 0.1 scale. 1=potential relative length of longest leaf

procedure RS\_EvalRelPotLeafLength(const NumPhase, HaunIndex, RankLongestLeaf : Double; var RelPotLeafLength : Double);

begin

try

if (NumPhase > 1) then

begin

RelPotLeafLength := Min((0.1 + 0.9 \* HaunIndex / RankLongestLeaf), 1);

end;

except

AfficheMessageErreur('RS\_EvalRelPotLeafLength', URisocas);

end;

end;

**Module n°51 - RS\_EvolPlantTilNumTot\_V2**

This module calculates tiller production as a function of the state variable Ic (current supply/demand ratio, driver of most adjustment processes in the model), the drought stress coefficient, square root of light interception (as a proxy of light quality effects) and the crop parameter TilAbility (between 0 and 1 usually):

Cstr is between 0 and 1(1 = stress free), Ic between 0 and >>1 (1 = source-sink are balanced). An additional parameter IcTillering sets the Ic below which tillering does not happen. It is strongly recommended not to modifiy this parameter from its default value O.5, unless you want to test specific hypotheses!

Tillering is only possible during NumPhase 2 (BVP) and 3 (PSP), and can only onset after HaunCritTillering leaves have appeared (therefore, if HaunIndex > HaunCritTillering).

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - ChangePhase** -IN- : ce booléen permet de savoir si la journée courante est une journée de changement de phase (facilite l'initialisation)

**3 - PlantsPerHill** -IN- : Number of seeds placed together in a hill (supposing all will germinate and grow)

**4 - TilAbility** -IN- (en fraction) : Sets capacity of plant to tiller if Ic > IcTillering. 0.3 gives already high tillering if conditions are favorable. Value 0 inhibits tillering

**5 - Density** -IN- (en pieds/Ha)

**6 - Ic** -IN- (en g/g) : state variable "index of competition" = daily assimilate supply/demand

**7 - IcTillering** -IN- (en none) : Value of Ic below which tillering cannot happen because of resource restrictions. Modify with caution

**8 - Cstr** -IN- (en none) : drought stress coefficient: FTSW is transformed into Cstr by FAO function using P-factor

**9 - HaunIndex** -IN- (en none) : Number of leaves appeared on main stem, including those that have already senesced

**10 - HaunCritTillering** -IN- (en none) : Leaf number on main culm above which tillering can happen. Usually 3 or 4

**11 - LTRkdfcl** -IN- (en fraction) : Light transmission rate of canopy as calculated with Kdfcl (taking into account crop Kdf and clumping), = 1-LIRkdfcl

**12 - CulmsPerHill** -INOUT-

**13 - CulmsPerPlant** -INOUT- (en till/plant) : Tiller number per plant (without main stem)

**14 - CulmsPop** -INOUT- (en till/ha) : Tiller number per ha (without main stem)

procedure RS\_EvolPlantTilNumTot\_V2(const NumPhase, ChangePhase, PlantsPerHill, TilAbility, Density, Ic, IcTillering, cstr, HaunIndex, HaunCritTillering, LtrKdfcl : Double; var CulmsPerHill, CulmsPerPlant, CulmsPop : Double);

var

TilNewPlant : Double;

begin

try

if ((NumPhase = 1) and (ChangePhase = 1)) then

begin

CulmsPerHill := PlantsPerHill;

end

else

begin

if ((NumPhase = 2) and (ChangePhase = 1)) then

begin

CulmsPerPlant := 1;

CulmsPop := CulmsPerPlant \* Density \* PlantsPerHill;

end

else

begin

if ((NumPhase > 1) and (NumPhase < 4) and (HaunIndex > HaunCritTillering)) then

begin

TilNewPlant := cstr \* Min(Max(0, (Ic - IcTillering) \* TilAbility)\* Sqrt(LtrKdfcl), 2);

CulmsPerPlant := CulmsPerPlant + TilNewPlant;

CulmsPerHill := CulmsPerPlant \* PlantsPerHill;

CulmsPop := CulmsPerHill \* Density;

end

else

begin

CulmsPerPlant := CulmsPerPlant;

CulmsPop := CulmsPop;

CulmsPerHill := CulmsPerHill;

end;

end;

end;

except

AfficheMessageErreur('RS\_EvolPlantTilNumTot\_V2', URisocas);

end;

end;

**Module n°52 - RS\_EvolPlantLeafNumTot**

This module calculates PlantLeafNumTot, the total leaf number produced on a plant hill (icluding leaves that have already died). Note that if there are several plants in a hill (parameter PlantsPerHill), individual plnts have a smaller total leaf number, and this information is not output. The total leaf number produced on the main culm is equal to the state variable HaunIndex. Both are output variables.Daily incremental leaf number production is equal to HaunGain \* CulmsPerHill. PlantLeafNumNew is accrued daily to give PlantLeafNumTot.

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - CulmsPerHill** -IN-

**3 - HaunGain** -IN-

**4 - PlantLeafNumNew** -INOUT-

**5 - PlantLeafNumTot** -INOUT- (en leave/plant) : Total number of leaves produced by plant, including green and dead

procedure RS\_EvolPlantLeafNumTot(const NumPhase, CulmsPerHill, HaunGain : Double; var PlantLeafNumNew, PlantLeafNumTot : Double);

begin

try

if ((NumPhase > 1) and (NumPhase < 5)) then

begin

PlantLeafNumNew := HaunGain \* CulmsPerHill;

PlantLeafNumTot := PlantLeafNumTot + PlantLeafNumNew;

end

else

begin

PlantLeafNumNew := PlantLeafNumNew;

PlantLeafNumTot := PlantLeafNumTot;

end;

except

AfficheMessageErreur('RS\_EvolPlantLeafNumTot', URisocas);

end;

end;

**Module n°53 - RS\_EvolMobiliTillerDeath\_V2**

This module calculates the number of tillers that die on a given day, based on the crop parameter CoeffTillerDeath (between 0 and 0.5, roughly) and competition index Ic. The dead tillers are subtracted from the total tiller number. Tillers can die anytime in NumPhase 3 (PSP) and 4 (RPR) except during the last 3rd of RPR, during which booting and heading happens and the surviving tillers are protected. This is an observation only made on rice (Dingkuhn et al.) but we generalize it here. Dry matter of duing tillers is assumed to be recycled in the plant. This may not be entirely true but the resulting error is small because aborted tillers are usually small.

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - SDJCorPhase4** -IN- (en °C.jour)

**3 - SDJRPR** -IN- (en °C.d) : Phase 4. Sets duration from PI to Flowering. Period of internode and panicle (structural component) development

**4 - CoeffTillerDeath** -IN- (en fraction) : Sets rate of tiller abortion (as fraction of existing number) provided Ic falls below 0

**5 - Density** -IN- (en pieds/Ha)

**6 - Ic** -IN- (en g/g) : state variable "index of competition" = daily assimilate supply/demand

**7 - PlantsPerHill** -IN- : Number of seeds placed together in a hill (supposing all will germinate and grow)

**8 - TillerDeathPop** -OUT- (en tiller/d/ha) : Daily number of senesced tillers per ha

**9 - CulmsPop** -INOUT- (en till/ha) : Tiller number per ha (without main stem)

**10 - CulmsPerPlant** -INOUT- (en till/plant) : Tiller number per plant (without main stem)

**11 - CulmsPerHill** -INOUT- (en none)

**12 - DryMatStructPaniclePop** -INOUT- (en kg/ha) : Panicle structural dry matter at population scale (does not include grains), formed between PI and flowering

procedure RS\_EvolMobiliTillerDeath\_V2(const NumPhase, SDJPhase4, SeuilTempRPR, CoeffTillerDeath, Density, Ic, PlantsPerHill : Double; var TillerDeathPop, CulmsPop, CulmsPerPlant, CulmsPerHill, DryMatStructPaniclePop : Double);

begin

try

if ((NumPhase = 3) or ((NumPhase = 4) and (SDJPhase4 <= 0.67 \* SeuilTempRPR)) and (CulmsPerPlant >= 1)) then

begin

TillerDeathPop := (1 - (Min(Ic, 1))) \* CulmsPop \* CoeffTillerDeath;

CulmsPop := CulmsPop - TillerDeathPop;

CulmsPerPlant := CulmsPop / (Density \* PlantsPerHill);

CulmsPerHill := CulmsPerPlant \* PlantsPerHill;

DryMatStructPaniclePop := DryMatStructPaniclePop \* Max(0, CulmsPop) / (CulmsPop + TillerDeathPop);

end;

except

AfficheMessageErreur('RS\_EvolMobiliTillerDeath\_V2', URisocas);

end;

end;

**Module n°54 - RS\_EvolMobiliLeafDeath**

This module calculates daily leaf death in terms of dry matter and leaf area, as a fraction of existing leaf mass, the competition index Ic and the crop parameter CoeffLeafDeath (0…0,5, roughly). The process is calculated summarily for the leaf compartment, without considering position. Leaves can die anytime during the entire crop cycle. A fraction of 0.2 of leaf dw is recycled into the daily assimilate pool, and 0.8 appear as dead leaf material (output variable DeadLeafDrywtPop). LaiDead is also simulated.

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - Ic** -IN- (en g/g) : state variable "index of competition" = daily assimilate supply/demand

**3 - CoeffLeafDeath** -IN- (en fraction) : Coefficient for leaf death sensitivity to resource restriction, function of Ic

**4 - Sla** -IN- (en ha/kg) : Specific leaf area (reciprocal of specific leaf weight). High values indicate thin leaves

**5 - LeafDeathPop** -OUT- (en kg/ha)

**6 - DryMatStructLeafPop** -INOUT- (en kg/ha) : Green leaf blade dry matter at population scale

**7 - MobiliLeafDeath** -OUT- (en kg/ha)

**8 - DeadLeafdrywtPop** -INOUT- (en kg/ha) : Dead leaf dry mass (assuming they do not decompose; but exluding the mass that has been recycled)

**9 - LaiDead** -INOUT- (en m²/m²) : Dead leaf area index, assuming they don't shrink nor decompose

procedure RS\_EvolMobiliLeafDeath(const NumPhase, Ic, CoeffLeafDeath, sla : Double; var LeafDeathPop, DryMatStructLeafPop, MobiliLeafDeath, DeadLeafDrywtPop, LaiDead : Double);

begin

try

if (NumPhase > 1) then

begin

LeafDeathPop := (1 - (Min(Ic, 1))) \* DryMatStructLeafPop \* CoeffLeafDeath;

DryMatStructLeafPop := DryMatStructLeafPop - LeafDeathPop;

MobiliLeafDeath := 0.2 \* LeafDeathPop;

DeadLeafDrywtPop := DeadLeafDrywtPop + (0.8 \* LeafDeathPop);

LaiDead := DeadLeafDrywtPop \* sla;

end;

except

AfficheMessageErreur('RS\_EvolMobiliLeafDeath', URisocas);

end;

end;

**Module n°55 - RS\_EvalSupplyTot**

This module calculates the daily assimilate pool (SupplyTot) available for growth, consisting of Assim + mobilizate from dead leaves – maintenance respiration – RespMaintDepth. RespMaintDepth is a carry-over from the previous day, for the rare case that maintenance cost is higher than assimilation.

As a next step, the RespMaintDepth of the current day is calculated, if there is any, in which case SupplyTot =0.

The next step is a bit complex: from the previous day, a quantity of AssimSurplus may be carried over, as a result of sink limitation. If there is no internode compartment available that might absorb this surplus as storage (this is simulated further down), the surplus is declared as “AssimNotUsed” and inventoried as a cumulative variable. It will never appear as dry matter on the plant and can be interpreted as either feedback inhibition of photosynthesis, or as luxury respiration loss. If there is an internode compartment available to store the AssimSurplus, it remains declared as such and will be used as simulated further down.

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - PhaseStemElongation** -IN- (en none) : Indicates whether internodes are elongating (1) or not (0)

**3 - Assim** -IN- (en kg/ha/d) : Assim=AssimPot \* Cstr (if applicable, corrected with CstrAssim)

**4 - MobiliLeafDeath** -IN- (en kg/ha)

**5 - RespMaintTot** -IN- (en kg/ha/d) : Total daily maintenance respiration (Rm), sum of that of all organs as calculated with organ specific coefficients

**6 - RespMaintDebt** -OUT- (en kg/ha) : Rm demand that cannot be satisfied with current supply is shifted as a debt to the next day

**7 - AssimNotUsed** -INOUT- (en kg/ha/d) : This assimilate is not used because all sinks and the reserve buffer are saturated

**8 - AssimNotUsedCum** -INOUT- (en kg/ha) : Accrued term of AssimNotUsed

**9 - AssimSurplus** -INOUT- (en kg/ha/d) : Daily assimilate surplus after allocation to structural growth and grain filling. This surplus goes into internode storage

**10 - SupplyTot** -OUT- (en kg/ha/d) : Net fresh assimilate supply per day = Assim-RespMaintTot

procedure RS\_EvalSupplyTot(const NumPhase, PhaseStemElongation, Assim, MobiliLeafDeath, RespMaintTot : Double; var RespMaintDebt, AssimNotUsed, AssimNotUsedCum, AssimSurplus, SupplyTot : Double);

begin

try

SupplyTot := Assim + MobiliLeafDeath - RespMaintTot - Max(0, RespMaintDebt);

if (SupplyTot <= 0) then

begin

RespMaintDebt := 0 - SupplyTot;

SupplyTot := 0;

end

else

begin

RespMaintDebt := 0;

end;

if ((NumPhase < 5) and (PhaseStemElongation = 0)) then

begin

AssimNotUsed := AssimSurplus;

AssimNotUsedCum := AssimNotUsedCum + AssimNotUsed;

end

else

begin

AssimNotUsed := 0;

AssimNotUsedCum := AssimNotUsedCum + AssimNotUsed;

end;

except

AfficheMessageErreur('RS\_EvalSupplyTot', URisocas);

end;

end;

**Module n°56 - RS\_EvalDemandStructLeaf\_V2**

This module calculates assimilate demand for leaf growth (considered as entirely structural in this version for simplicity; only internodes and grains contain storage in this model!). Demand is calculated on a leaf area basis, and only thereafter converted to dry matter demand by dividing it by SLA. The leaf area demand is the product of potential individual leaf area (= squared potential leaf length \* parameter width/length ratio \* allometric coefficient 0.725), number of new leaves per plant, and stress coefficient Cstr (which is assumed to reduce area expansion linearly). All coefficients of the type 1000000, 0.1 etc. are just there to take care of unit conversions.

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - PlantLeafNumNew** -IN-

**3 - SlaNew** -IN- (en kg/ha)

**4 - SlaMax** -IN- (en kg/ha) : Initial (maximal) value of SLA (leaf surface/dw) for bulk canopy

**5 - RelPotLeafLength** -IN- (en fraction) : Relative length of leaf blades currently developing, or the last one that developed, on a 0.1 scale. 1=potential relative length of longest leaf

**6 - Density** -IN- (en pieds/Ha)

**7 - LeafLengthMax** -IN- (en mm) : Maximal individual length of the longest leaf blade (may not be attainted if constraints)

**8 - CoeffLeafWLRatio** -IN- (en fraction) : Maximal leaf blade width as fraction of length

**9 - Cstr** -IN- (en none) : drought stress coefficient: FTSW is transformed into Cstr by FAO function using P-factor

**10 - StressCold** -IN- (en Coeff x)

**11 - DemLeafAreaPlant** -INOUT-

**12 - DemStructLeafPlant** -INOUT-

**13 - DemStructLeafPop** -INOUT-

procedure RS\_EvalDemandStructLeaf\_V2(const NumPhase, PlantLeafNumNew, SlaNew, SlaMax, RelPotLeafLength, Density, LeafLengthMax, CoeffLeafWLRatio, cstr, StressCold : Double; var DemLeafAreaPlant, DemStructLeafPlant, DemStructLeafPop : Double);

var

CorrectedSla : Double;

begin

try

if ((NumPhase > 1) and (NumPhase < 5)) then

begin

DemLeafAreaPlant := (Power((RelPotLeafLength \* LeafLengthMax),2) \* CoeffLeafWLRatio \* 0.725 \* PlantLeafNumNew / 1000000) \* Min(cstr, StressCold);

if (SlaNew = 0) then

begin

CorrectedSla := SlaMax;

end

else

begin

CorrectedSla := SlaNew;

end;

DemStructLeafPlant := DemLeafAreaPlant \* 0.1 / CorrectedSla;

DemStructLeafPop := DemStructLeafPlant \* Density / 1000;

end;

except

AfficheMessageErreur('RS\_EvalDemandStructLeaf\_V2', URisocas);

end;

end;

**Module n°57 - RS\_EvalDemandStructSheath**

This module calculateds assimilate demand for leaf sheath growth (considered as entirely structural in this version for simplicity; only internodes and grains contain storage in this model!). It is assumed to be proportional to leaf blade demand on the basis of an allometric parameter WtRatioLeafSheath.But during early stages, sheath demand is downsized with an empirical function on the basis of SLA (just taking advantage of the fact that SLA decreases during early stages and then levels off). The result is an initially reduced sheath demand by half, which gives the plant an early growth boost. Without this correction, the leaf/shoot assimilate partitioning ratio would show a plateau grom germination to PI, whereas it is known to decrease steadily. With the present algorithm, this trend is achieved while fully maintaining the supply-demand concept that is absent in rigid partitioning models.

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - DemStructLeafPop** -IN-

**3 - WtRatioLeafSheath** -IN- (en fraction)

**4 - SlaMin** -IN- (en kg/ha) : Final (minimal) value of SLA (leaf surface/dw) for bulk canopy

**5 - SlaMax** -IN- (en kg/ha) : Initial (maximal) value of SLA (leaf surface/dw) for bulk canopy

**6 - Sla** -IN- (en ha/kg) : Specific leaf area (reciprocal of specific leaf weight). High values indicate thin leaves

**7 - StressCold** -IN- (en Coeff x)

**8 - DemStructSheathPop** -OUT-

procedure RS\_EvalDemandStructSheath(const NumPhase, DemStructLeafPop, WtRatioLeafSheath, SlaMin, SlaMax, Sla, StressCold : Double; var DemStructSheathPop : Double);

begin

try

if ((NumPhase > 1) and (NumPhase < 5)) then

begin

DemStructSheathPop := (1 + ((SlaMax - Sla) / (SlaMax - SlaMin))) \*

0.5 \* DemStructLeafPop / WtRatioLeafSheath \*

Max(0.00001, StressCold);

end;

except

AfficheMessageErreur('RS\_EvalDemandStructSheath', URisocas);

end;

end;

**Module n°58 - RS\_EvalDemandStructRoot\_V2**

This module calculates assimilate demand for root growth (**DemStructRootPop** ) on the basis of soil volume occupied by the root system (**RootSystVolPop**), the crop parameter setting the maximal root dry matter per soil volume (**CoeffRootMassPerVolMax**) and a partitioning coefficient (**RootPartitMax**) that sets the maximal root demand relative to leaf+sheath demand. **RootSystVolPop** is calculated as the rootfront depth to the 3rd power (a cube) if plant spacing permits it, otherwise it is laterally limited by spacing. Consequently, plants grown at high population density have less demand for growth than plants grown widely spaced..

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - Density** -IN- (en pieds/Ha)

**3 - CoeffRootMassPerVolMax** -IN- (en kg/m3) : Maximal root dry weight that can be produced per cubic meter of soil explored by root system. Sets demand for root partitioning, resulting value

**4 - RootPartitMax** -IN- (en g/g) : Upper limit of daily incremental assimilate partition to roots. Value 0.5 is a good default value

**5 - GrowthStructTotPop** -IN-

**6 - RootFront** -IN- (en mm) : depth of root front

**7 - SupplyTot** -IN- (en kg/ha/d) : Net fresh assimilate supply per day = Assim-RespMaintTot

**8 - DemStructLeafPop** -IN-

**9 - DemStructSheathPop** -IN-

**10 - DryMatStructRootPop** -IN- (en kg/ha) : Root blade dry matter at population scale

**11 - RootSystSoilSurfPop** -OUT- (en m2)

**12 - RootSystVolPop** -OUT- (en m3)

**13 - GainRootSystVolPop** -OUT- (en m3)

**14 - GainRootSystSoilSurfPop** -OUT- (en m2)

**15 - DemStructRootPop** -OUT-

**16 - RootSystSoilSurfPopOld** -INOUT-

**17 - RootFrontOld** -INOUT- (en mm)

**18 - RootSystVolPopOld** -INOUT- (en m3)

**19 - DemStructRootPlant** -OUT-

procedure RS\_EvalDemandStructRoot\_V2(const NumPhase, Density : Double; CoeffRootMassPerVolMax, RootPartitMax, GrowthStructTotPop, RootFront, SupplyTot, DemStructLeafPop, DemStructSheathPop, DryMatStructRootPop : Double; var RootSystSoilSurfPop, RootSystVolPop, GainRootSystVolPop, GainRootSystSoilSurfPop, DemStructRootPop, RootSystSoilSurfPopOld, RootFrontOld, RootSystVolPopOld, DemStructRootPlant: Double);

begin

try

RootSystSoilSurfPop := Min(RootFront \* RootFront \* Density / 1000000, 10000);

RootSystVolPop := RootSystSoilSurfPop \* RootFront / 1000;

if ((NumPhase > 1) and (NumPhase < 5)) then

begin

GainRootSystSoilSurfPop := RootSystSoilSurfPop - RootSystSoilSurfPopOld;

GainRootSystVolPop := RootSystVolPop - RootSystVolPopOld;

DemStructRootPop := Min((DemStructLeafPop + DemStructSheathPop) \*

RootPartitMax, Max(0, CoeffRootMassPerVolMax \*

RootSystVolPop - DryMatStructRootPop));

DemStructRootPlant := DemStructRootPop \* 1000 / density;

RootSystSoilSurfPopOld := RootSystSoilSurfPop;

RootFrontOld := RootFront;

RootSystVolPopOld := RootSystVolPop;

end;

except

AfficheMessageErreur('RS\_EvalDemandStructRoot\_V2', URisocas);

end;

end;

**Module n°59 - RS\_EvalDemandStructIN\_V2**

This module calculates the demand for assimilates for internode growth, based on incremental elongation (ApexHeightGain), culm number (CulmsPerHill), a crop parameter setting the potential internode dry weight per length (CoeffInternodeMass) and the competition index Ic (here set as a limiting factor between 0 and 1, applied as square root to achieve a progressive effect). Note that this demand is only for structural mass and does not include internode reserve storage that is calculated elsewhere. The module is only implemented during internode elongation that ends at flowering.

**1 - PhaseStemElongation** -IN- (en none) : Indicates whether internodes are elongating (1) or not (0)

**2 - ApexHeightGain** -IN- (en mm)

**3 - CulmsPerHill** -IN-

**4 - CoeffInternodeMass** -IN- (en g/mm) : Maximal structural mass of internode per mm length

**5 - Density** -IN- (en pieds/Ha)

**6 - Ic** -IN- (en g/g) : state variable "index of competition" = daily assimilate supply/demand

**7 - DemStructInternodePlant** -OUT-

**8 - DemStructInternodePop** -OUT-

procedure RS\_EvalDemandStructIN\_V2(const PhaseElongation, ApexHeightGain, CulmsPerHill, CoeffInternodeMass, Density, Ic : Double; var DemStructInternodePlant, DemStructInternodePop : Double);

begin

try

if (PhaseElongation = 1) then

begin

DemStructInternodePlant := Power(Min(Ic, 1), 0.5) \*

ApexHeightGain \* CulmsPerHill \* CoeffInternodeMass;

DemStructInternodePop := DemStructInternodePlant \* Density / 1000;

end;

except

AfficheMessageErreur('RS\_EvalDemandStructIN\_V2', URisocas);

end;

end;

**Module n°60 - RS\_EvalDemandStructPanicle\_V2**

This module calculates the assimilate demand of the structural part of the panicle during its development, between PI and flowering (NumPhase 4). Demand equals the product of the parameter CoeffPanicleMass (setting the structural growth rate of the panicle a,d thus, indirectly, the potential harvest index), culm number (CulmsPerHill) and the competition index Ic. This structural growth is stopped (and demand set to zero) when the accumulated structural mass exceeds potential panicle structural weight (parameter PanStructMassMax). This permits to implement a genetic limitation to panicle size.

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - CoeffPanicleMass** -IN- (en none) : Sets growth rate of structural parts of panicle between PI and flowering, subject to limitation by ressource availability and genetic size limit

**3 - CulmsPerHill** -IN-

**4 - Ic** -IN- (en g/g) : state variable "index of competition" = daily assimilate supply/demand

**5 - DryMatStructPaniclePop** -IN- (en kg/ha) : Panicle structural dry matter at population scale (does not include grains), formed between PI and flowering

**6 - Density** -IN- (en pieds/Ha)

**7 - PanStructMassMax** -IN- (en g) : Upper limit of individual panicle mass (structural parts only including peduncle)

**8 - StressCold** -IN- (en Coeff x)

**9 - DemStructPaniclePlant** -OUT-

**10 - PanStructMass** -OUT-

**11 - DemStructPaniclePop** -OUT-

procedure RS\_EvalDemandStructPanicle\_V2(const NumPhase, CoeffPanicleMass, CulmsPerHill, Ic, DryMatStructPaniclePop, Density, PanStructMassMax, StressCold : Double; var DemStructPaniclePlant, PanStructMass, DemStructPaniclePop : Double);

begin

try

if (NumPhase = 4) then

begin

DemStructPaniclePlant := CoeffPanicleMass \* CulmsPerHill \*

Sqrt(Min(Ic, 1)) \* Sqrt(Max(0.00001, StressCold));

PanStructMass := 1000 \* DryMatStructPaniclePop / (Density \* CulmsPerHill);

if (PanStructMass > PanStructMassMax) then

begin

DemStructPaniclePlant := 0;

end;

DemStructPaniclePop := DemStructPaniclePlant \* Density / 1000;

end;

except

AfficheMessageErreur('RS\_EvalDemandStructPanicle\_V2', URisocas);

end;

end;

**Module n°61 - RS\_EvalDemandTotAndIcPreFlow**

This module calculates the internal competition index Ic (supply/demanda t the plant scale, in this module only grom germination to flowering (Ic for ripening stages is calculated elsewhere). For this purpose, aggregate assimilate demand for the different organs is calculated, and Ic is calculated as SupplyTot/DemStrctTotPop. Lastly, a cumulativeIc and a floating mean Ic are calculated (on the basis of Ic truncated 0…1) for use in other modules.

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - RespMaintTot** -IN- (en kg/ha/d) : Total daily maintenance respiration (Rm), sum of that of all organs as calculated with organ specific coefficients

**3 - DemStructLeafPop** -IN-

**4 - DemStructSheathPop** -IN-

**5 - DemStructRootPop** -IN-

**6 - DemStructInternodePop** -IN-

**7 - DemStructPaniclePop** -IN-

**8 - SupplyTot** -IN- (en kg/ha/d) : Net fresh assimilate supply per day = Assim-RespMaintTot

**9 - NbDaysSinceGermination** -IN-

**10 - PlantHeight** -IN- (en mm) : Overall height of plant incuding top leaves, assuming vertical orientation

**11 - Cstr** -IN- (en none) : drought stress coefficient: FTSW is transformed into Cstr by FAO function using P-factor

**12 - DemStructTotPop** -OUT-

**13 - Ic** -INOUT- (en g/g) : state variable "index of competition" = daily assimilate supply/demand

**14 - IcCum** -INOUT- (en kg/kg)

**15 - IcMean** -OUT- (en none) : Accued mean of Ic

**16 - CstrCum** -INOUT- (en none)

**17 - CstrMean** -OUT- (en none)

procedure RS\_EvalDemandTotAndIcPreFlow(const NumPhase, RespMaintTot, DemStructLeafPop, DemStructSheathPop, DemStructRootPop, DemStructInternodePop, DemStructPaniclePop, SupplyTot, NbDaysSinceGermination, PlantHeight, Cstr : Double; var DemStructTotPop, Ic, IcCumul, IcMean, CstrCumul, CstrMean : Double);

begin

try

if ((NumPhase > 1) and (NumPhase < 5)) then

begin

DemStructTotPop := DemStructLeafPop + DemStructSheathPop + DemStructRootPop + DemStructInternodePop + DemStructPaniclePop;

Ic := SupplyTot / DemStructTotPop;

if (Ic <= 0) then

begin

Ic := 0;

end;

if (PlantHeight = 0) then

begin

Ic := 1;

end;

IcCumul := IcCumul + Min(Ic, 1);

IcMean := IcCumul / NbDaysSinceGermination;

CstrCumul := CstrCumul + Cstr;

CstrMean := CstrCumul / NbDaysSinceGermination;

end;

if ((NumPhase = 5) or (NumPhase = 6)) then

begin

IcCumul := IcCumul + Min(Ic, 1);

IcMean := IcCumul / NbDaysSinceGermination;

CstrCumul := CstrCumul + Cstr;

CstrMean := CstrCumul / NbDaysSinceGermination;

end;

except

AfficheMessageErreur('RS\_EvalDemandTotAndIcPreFlow', URisocas);

end;

end;

**Module n°62 - RS\_EvolGrowthStructLeafPop**

This module calculates leaf growth based on its demand, adjusted to the available resources. Growth cannot be greater than demand, so it is possible that some of the assimilates will not be used.

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - Ic** -IN- (en g/g) : state variable "index of competition" = daily assimilate supply/demand

**3 - SupplyTot** -IN- (en kg/ha/d) : Net fresh assimilate supply per day = Assim-RespMaintTot

**4 - DemStructLeafPop** -IN-

**5 - DemStructTotPop** -IN-

**6 - GrowthStructLeafPop** -OUT-

procedure RS\_EvolGrowthStructLeafPop(const NumPhase, Ic, SupplyTot, DemStructLeafPop, DemStructTotPop : Double; var GrowthStructLeafPop : Double);

begin

try

if ((NumPhase > 1) and (NumPhase < 5)) then

begin

if (Ic < 1) then

begin

GrowthStructLeafPop := SupplyTot \* (DemStructLeafPop / DemStructTotPop);

end

else

begin

GrowthStructLeafPop := DemStructLeafPop;

end;

end;

except

AfficheMessageErreur('RS\_EvolGrowthStructLeafPop', URisocas);

end;

end;

**Module n°63 - RS\_EvolGrowthStructSheathPop**

This module calculates sheath growth based on its demand, adjusted to the available resources. Growth cannot be greater than demand, so it is possible that some of the assimilates will not be used.

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - Ic** -IN- (en g/g) : state variable "index of competition" = daily assimilate supply/demand

**3 - SupplyTot** -IN- (en kg/ha/d) : Net fresh assimilate supply per day = Assim-RespMaintTot

**4 - DemStructSheathPop** -IN-

**5 - DemStructTotPop** -IN-

**6 - GrowthStructSheathPop** -OUT-

procedure RS\_EvolGrowthStructSheathPop(const NumPhase, Ic, SupplyTot, DemStructSheathPop, DemStructTotPop : Double; var GrowthStructSheathPop : Double);

begin

try

if ((NumPhase > 1) and (NumPhase < 5)) then

begin

if (Ic < 1) then

begin

GrowthStructSheathPop := SupplyTot \* (DemStructSheathPop / DemStructTotPop);

end

else

begin

GrowthStructSheathPop := DemStructSheathPop;

end;

end;

except

AfficheMessageErreur('RS\_EvolGrowthStructSheathPop', URisocas);

end;

end;

**Module n°64 - RS\_EvolGrowthStructRootPop**

This module calculates root growth based on its demand, adjusted to the available resources. Growth cannot be greater than demand, so it is possible that some of the assimilates will not be used.

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - Ic** -IN- (en g/g) : state variable "index of competition" = daily assimilate supply/demand

**3 - SupplyTot** -IN- (en kg/ha/d) : Net fresh assimilate supply per day = Assim-RespMaintTot

**4 - DemStructRootPop** -IN-

**5 - DemStructTotPop** -IN-

**6 - GrowthStructRootPop** -OUT-

procedure RS\_EvolGrowthStructRootPop(const NumPhase, Ic, SupplyTot, DemStructRootPop, DemStructTotPop : Double; var GrowthStructRootPop : Double);

begin

try

if ((NumPhase > 1) and (NumPhase < 5)) then

begin

if (Ic < 1) then

begin

GrowthStructRootPop := SupplyTot \* (DemStructRootPop / DemStructTotPop);

end

else

begin

GrowthStructRootPop := DemStructRootPop;

end;

end;

except

AfficheMessageErreur('RS\_EvolGrowthStructRootPop', URisocas);

end;

end;

**Module n°65 - RS\_EvolGrowthStructINPop**

This module calculates internode (structural) growth based on its demand, adjusted to the available resources. Growth cannot be greater than demand, so it is possible that some of the assimilates will not be used.

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - Ic** -IN- (en g/g) : state variable "index of competition" = daily assimilate supply/demand

**3 - SupplyTot** -IN- (en kg/ha/d) : Net fresh assimilate supply per day = Assim-RespMaintTot

**4 - DemStructInternodePop** -IN-

**5 - DemStructTotPop** -IN-

**6 - GrowthStructInternodePop** -OUT-

procedure RS\_EvolGrowthStructINPop(const NumPhase, Ic, SupplyTot, DemStructInternodePop, DemStructTotPop : Double; var GrowthStructInternodePop : Double);

begin

try

if ((NumPhase > 1) and (NumPhase < 5)) then

begin

if (Ic < 1) then

begin

GrowthStructInternodePop := SupplyTot \* (DemStructInternodePop / DemStructTotPop);

end

else

begin

GrowthStructInternodePop := DemStructInternodePop;

end;

end;

except

AfficheMessageErreur('RS\_EvolGrowthStructInternodePop', URisocas);

end;

end;

**Module n°66 - RS\_EvolGrowthStructPanPop**

This module calculates panicle (structural) growth based on its demand, adjusted to the available resources. Growth cannot be greater than demand, so it is possible that some of the assimilates will not be used.

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - Ic** -IN- (en g/g) : state variable "index of competition" = daily assimilate supply/demand

**3 - SupplyTot** -IN- (en kg/ha/d) : Net fresh assimilate supply per day = Assim-RespMaintTot

**4 - DemStructPaniclePop** -IN-

**5 - DemStructTotPop** -IN-

**6 - GrowthStructPaniclePop** -OUT-

procedure RS\_EvolGrowthStructPanPop(const NumPhase, Ic, SupplyTot, DemStructPaniclePop, DemStructTotPop : Double; var GrowthStructPaniclePop : Double);

begin

try

if ((NumPhase > 1) and (NumPhase < 5)) then

begin

if (Ic < 1) then

begin

GrowthStructPaniclePop := SupplyTot \* (DemStructPaniclePop / DemStructTotPop);

end

else

begin

GrowthStructPaniclePop := DemStructPaniclePop;

end;

end;

except

AfficheMessageErreur('RS\_EvolGrowthStructPaniclePop', URisocas);

end;

end;

**Module n°67 - RS\_Priority2GrowthPanStrctPop**

This module permits attributing priority to panicke structural development as compared to stem elongation (a high-cost process) during the period from PI to flowering (NumPhase 4). This way, the plant protects its sink potential development even under conditions of fierce competition for assimilates during stem elongation, for example when population density is high, tiller number is high or plants are tall. Value 0 = eaqual priority to panicle and internode growth; 1 = full priority to panicle (within the limits of its demand); default value = 0.5

**1 - PriorityPan** -IN- (en Coeff x) : Priority given to panicle structural growth (0=normal, 1=max)

**2 - DemStructPaniclePop** -IN-

**3 - GrowthStructPaniclePop** -INOUT-

**4 - GrowthStructInternodePop** -INOUT-

procedure RS\_Priority2GrowthPanStrctPop(const PriorityPan, DemStructPaniclePop : Double; var GrowthStructPaniclePop, GrowthStructInternodePop : Double);

var

GrowthPanDeficit : Double;

begin

try

if (GrowthStructPaniclePop < DemStructPaniclePop) then

begin

GrowthPanDeficit := DemStructPaniclePop - GrowthStructPaniclePop;

GrowthStructPaniclePop := GrowthStructPaniclePop + Min(PriorityPan \* GrowthPanDeficit, GrowthStructInternodePop);

GrowthStructInternodePop := Max(0, GrowthStructInternodePop - PriorityPan \* GrowthPanDeficit);

end;

except

AfficheMessageErreur('RS\_Priority2GrowthPanStrctPop', URisocas);

end;

end;

**Module n°68 - RS\_EvolGrowthStructTot**

This module calculates total structural growth as the sum of growth of different organ classes. In the case of GrowthStructTotPop < SupplyTot, an assimilate surplus is calculated.

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - GrowthStructLeafPop** -IN-

**3 - GrowthStructSheathPop** -IN-

**4 - GrowthStructRootPop** -IN-

**5 - GrowthStructInternodePop** -IN-

**6 - GrowthStructPaniclePop** -IN-

**7 - SupplyTot** -IN- (en kg/ha/d) : Net fresh assimilate supply per day = Assim-RespMaintTot

**8 - GrowthStructTotPop** -OUT-

**9 - AssimSurplus** -INOUT- (en kg/ha/d) : Daily assimilate surplus after allocation to structural growth and grain filling. This surplus goes into internode storage

procedure RS\_EvolGrowthStructTot(const NumPhase, GrowthStructLeafPop, GrowthStructSheathPop, GrowthStructRootPop, GrowthStructInternodePop, GrowthStructPaniclePop, SupplyTot : Double; var GrowthStructTotPop, AssimSurplus : Double);

begin

try

if ((NumPhase > 1) and (NumPhase < 5)) then

begin

GrowthStructTotPop := GrowthStructLeafPop + GrowthStructSheathPop + GrowthStructRootPop + GrowthStructInternodePop + GrowthStructPaniclePop;

AssimSurplus := Max((SupplyTot - GrowthStructTotPop), 0);

end;

except

AfficheMessageErreur('RS\_EvolGrowthStructTot', URisocas);

end;

end;

**Module n°69 - RS\_AddResToGrowthStructPop**

This module calculates reserve mobilization from the internode reserve compartment if Ic<1 (thus, growth of organs was inferiour to demand). First, the potential amount of reserves that can be mobilized on that day is calculated based on parameter “RelMobiliInternodeMax “(fraction of current size of reserve compartment). Then structural growth deficit is determined. The mobilizable reserves (up to the amount needed) are then distributed among organs proportionally to their demand.After this, the demand is either satisfied and some reserves may be left in storage, or a deficit remains, resulting in sub-maximal growth.

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - Ic** -IN- (en g/g) : state variable "index of competition" = daily assimilate supply/demand

**3 - PhaseStemElongation** -IN- (en none) : Indicates whether internodes are elongating (1) or not (0)

**4 - DryMatResInternodePop** -IN-

**5 - DemStructTotPop** -IN-

**6 - RelMobiliInternodeMax** -IN- (en fraction) : Fraction of currently stored reserves in internodes that can be mobilized in one day, provided there is demand for it (Ic<1)

**7 - ResInternodeMobiliDayPot** -OUT-

**8 - GrowthStructDeficit** -OUT-

**9 - GrowthStructLeafPop** -INOUT-

**10 - GrowthStructSheathPop** -INOUT-

**11 - GrowthStructRootPop** -INOUT-

**12 - GrowthStructInternodePop** -INOUT-

**13 - GrowthStructPaniclePop** -INOUT-

**14 - GrowthStructTotPop** -INOUT-

**15 - ResInternodeMobiliDay** -OUT- (en kg/ha) : Daily rate of internode reserve mobilization

procedure RS\_AddResToGrowthStructPop(const NumPhase, Ic, PhaseStemElongation,

DryMatResInternodePop, DemStructTotPop, DemStructLeafPop, DemStructSheathPop,

DemStructRootPop, DemStructInternodePop, DemStructPaniclePop, RelMobiliInternodeMax: Double;

var ResInternodeMobiliDayPot, GrowthStructDeficit, GrowthStructLeafPop,

GrowthStructSheathPop, GrowthStructRootPop, GrowthStructInternodePop,

GrowthStructPaniclePop, GrowthStructTotPop, ResInternodeMobiliDay: Double);

begin

try

if (NumPhase > 1) then

begin

if (NumPhase > 2) then

begin

ResInternodeMobiliDayPot := RelMobiliInternodeMax \*

DryMatResInternodePop;

GrowthStructDeficit := Max((DemStructTotPop - GrowthStructTotPop), 0);

end;

if ((Ic < 1) and (DemStructTotPop > 0)) then

begin

ResInternodeMobiliDay := Min(ResInternodeMobiliDayPot,

GrowthStructDeficit);

GrowthStructTotPop := GrowthStructLeafPop + GrowthStructSheathPop

+ GrowthStructRootPop + GrowthStructInternodePop +

GrowthStructPaniclePop;

GrowthStructLeafPop := GrowthStructLeafPop + ResInternodeMobiliDay \*

(DemStructLeafPop / DemStructTotPop);

GrowthStructSheathPop := GrowthStructSheathPop + ResInternodeMobiliDay \*

(DemStructSheathPop / DemStructTotPop);

GrowthStructRootPop := GrowthStructRootPop + ResInternodeMobiliDay \*

(DemStructRootPop / DemStructTotPop);

GrowthStructInternodePop := GrowthStructInternodePop +

ResInternodeMobiliDay \* (DemStructInternodePop / DemStructTotPop);

GrowthStructPaniclePop := GrowthStructPaniclePop + ResInternodeMobiliDay

\* (DemStructPaniclePop / DemStructTotPop);

GrowthStructTotPop := GrowthStructLeafPop + GrowthStructSheathPop

+ GrowthStructRootPop + GrowthStructInternodePop +

GrowthStructPaniclePop;

end;

end;

except

AfficheMessageErreur('RS\_AddResToGrowthStructPop' +

' GrowthStrucTotPop : ' + floattostr(GrowthStructTotPop), URisocas);

end;

end;

**Module n°70 - RS\_EvolDemPanFilPopAndIcPFlow**

This module calculates demand of the panicle for filling and recalculates Ic. A separate routine for calculating Ic is necessary at post-floral stages because at that time, all structural growth is over and the only assimilate-consuming processes are panicle filling and maintenance respiration. Panicle demand for filling of PanicleSinkPop which is proportional to the accumulated structural mass of the panicle before flowering, multiplied by CoeffPanicleSink, and the sterility fraction removed. Panicle filling ends at NumPhase 6 (Matu2), and during this last phase maintenance respiration is the only sink for assimilates. Throughout these processes, internodes can store or mobilize reserves, buffering sink-source imbalances.

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - DryMatStructPaniclePop** -IN- (en kg/ha) : Panicle structural dry matter at population scale (does not include grains), formed between PI and flowering

**3 - CoeffPanSinkPop** -IN- (en fraction) : Sets the grain mass (yield) that can be produced per structural mass of panicle including peduncle

**4 - SterilityTot** -IN- (en fraction) : Total spikelet sterility (caused by cold, heat and drought)

**5 - DegresDuJourCor** -IN- (en °C.d) : same, but adjusted for drought effect using a value >0 for DEVcstr: drought slows development, thus reducing the effective heat dose available

**6 - SDJMatu1** -IN- (en °C.d) : Phase 5. Sets duration from flowering to end of grain filling. No more structural growth happens

**7 - SupplyTot** -IN- (en kg/ha/d) : Net fresh assimilate supply per day = Assim-RespMaintTot

**8 - Assim** -IN- (en kg/ha/d) : Assim=AssimPot \* Cstr (if applicable, corrected with CstrAssim)

**9 - RespMaintTot** -IN- (en kg/ha/d) : Total daily maintenance respiration (Rm), sum of that of all organs as calculated with organ specific coefficients

**10 - StressCold** -IN- (en Coeff x)

**11 - PanicleSinkPop** -OUT-

**12 - DemPanicleFillPop** -OUT-

**13 - AssimSurplus** -INOUT- (en kg/ha/d) : Daily assimilate surplus after allocation to structural growth and grain filling. This surplus goes into internode storage

**14 - Ic** -INOUT- (en g/g) : state variable "index of competition" = daily assimilate supply/demand

procedure RS\_EvolDemPanFilPopAndIcPFlow(const NumPhase, DryMatStructPaniclePop, CoeffPanSinkPop, SterilityTot, DegresDuJourCor, DegresNumPhase5, SupplyTot, Assim, RespMaintTot, StressCold : Double; var PanicleSinkPop, DemPanicleFillPop, AssimSurplus, Ic : Double);

begin

try

if (NumPhase = 5) then

begin

PanicleSinkPop := DryMatStructPaniclePop \* CoeffPanSinkPop \* (1 - SterilityTot);

DemPanicleFillPop := (DegresDuJourCor / DegresNumPhase5) \* PanicleSinkPop \*

Sqrt(Max(0.00001, StressCold));

Ic := SupplyTot / Max(DemPanicleFillPop, 0.0000001);

if (Ic <= 0) then

begin

Ic := 0;

end;

end;

if (NumPhase = 6) then

begin

Ic := Assim / RespMaintTot;

if (Ic >= 1) then

begin

AssimSurplus := Max(0, Assim - RespMaintTot);

end

else

begin

AssimSurplus := 0;

end;

if (Ic < 0) then

begin

Ic := 0;

end;

end;

except

AfficheMessageErreur('RS\_EvolDemPanFilPopAndIcPFlow', URisocas);

end;

end;

**Module n°71 - RS\_EvolPanicleFilPop**

This module implements the panicle demand for filling, based on current SupplyTot and internode reserves. Grain yield is calculated at the end of the module, as an evolving entity.

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - Ic** -IN- (en g/g) : state variable "index of competition" = daily assimilate supply/demand

**3 - DryMatResInternodePop** -IN-

**4 - DemPanicleFillPop** -IN-

**5 - SupplyTot** -IN- (en kg/ha/d) : Net fresh assimilate supply per day = Assim-RespMaintTot

**6 - RelMobiliInternodeMax** -IN- (en fraction) : Fraction of currently stored reserves in internodes that can be mobilized in one day, provided there is demand for it (Ic<1)

**7 - RespMaintTot** -IN- (en kg/ha/d) : Total daily maintenance respiration (Rm), sum of that of all organs as calculated with organ specific coefficients

**8 - Assim** -IN- (en kg/ha/d) : Assim=AssimPot \* Cstr (if applicable, corrected with CstrAssim)

**9 - ResInternodeMobiliDayPot** -OUT-

**10 - AssimSurplus** -INOUT- (en kg/ha/d) : Daily assimilate surplus after allocation to structural growth and grain filling. This surplus goes into internode storage

**11 - PanicleFilDeficit** -OUT-

**12 - ResInternodeMobiliDay** -OUT- (en kg/ha) : Daily rate of internode reserve mobilization

**13 - PanicleFilPop** -OUT-

**14 - GrainYieldPop** -INOUT- (en kg/ha) : Grain yield at population scale (without structural parts of panicle)

procedure RS\_EvolPanicleFilPop(const NumPhase, Ic, DryMatResInternodePop, DemPanicleFilPop, SupplyTot, RelMobiliInternodeMax, RespMaintTot, Assim : Double; var ResInternodeMobiliDayPot, AssimSurplus, PanicleFilDeficit, ResInternodeMobiliDay, PanicleFilPop, GrainYieldPop : Double);

begin

try

if (NumPhase = 5) then

begin

ResInternodeMobiliDayPot := RelMobiliInternodeMax \* DryMatResInternodePop;

if (Ic > 1) then

begin

PanicleFilPop := Max(DemPanicleFilPop, 0);

PanicleFilDeficit := 0;

AssimSurplus := SupplyTot - PanicleFilPop;

end

else

begin

if (Ic <= 1) then

begin

PanicleFilDeficit := Max((DemPanicleFilPop - Max(SupplyTot, 0)), 0);

ResInternodeMobiliDay := Max(Min(ResInternodeMobiliDayPot, 0.5 \* PanicleFilDeficit), 0);

PanicleFilPop := Max((SupplyTot + ResInternodeMobiliDay), 0);

AssimSurplus := 0;

end;

end;

GrainYieldPop := GrainYieldPop + PanicleFilPop;

end

else

begin

if (NumPhase = 6) then

begin

AssimSurplus := Assim - RespMaintTot;

ResInternodeMobiliDay := Min(Max(0, RespMaintTot - Assim), DryMatResInternodePop);

end

else

begin

if (NumPhase > 6) then

begin

ResInternodeMobiliDay := 0;

end;

end;

end;

except

AfficheMessageErreur('RS\_EvolPanicleFilPop', URisocas);

end;

end;

**Module n°72 - RS\_EvolGrowthReserveInternode**

This module updates the status of the internode reserve compartment (Attention: this compartment comprises now internode AND sheath reserves!) based on the day’s new storgae and mobilization. Excess assimilates that find no space in the reserve compartment are declared as “**AssimNotUsed”** and represent in the balance a feed-back inhibition of photosynthesis/ The cumulative of this unused quantity is calculated and output.

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - PhaseStemElongation** -IN- (en none) : Indicates whether internodes are elongating (1) or not (0)

**3 - DryMatStructInternodePop** -IN- (en kg/ha) : Internode blade dry matter at population scale (only structural component: reserves are simulated and output separately)

**4 - CoeffResCapacityInternode** -IN- (en fraction) : Sets upper limit of internode storage capacity, as fraction of current structural internode mass

**5 - AssimSurplus** -IN- (en kg/ha/d) : Daily assimilate surplus after allocation to structural growth and grain filling. This surplus goes into internode storage

**6 - ResInternodeMobiliDay** -IN- (en kg/ha) : Daily rate of internode reserve mobilization

**7 - ResCapacityInternodePop** -OUT- (en kg/ha) : Size of potential reservoir for reserves in internodes per ha

**8 - IncreaseResInternodePop** -OUT-

**9 - DryMatResInternodePop** -INOUT-

**10 - AssimNotUsed** -INOUT- (en kg/ha/d) : This assimilate is not used because all sinks and the reserve buffer are saturated

**11 - AssimNotUsedCum** -INOUT- (en kg/ha) : Accrued term of AssimNotUsed

**12 - GrowthResInternodePop** -OUT-

procedure RS\_EvolGrowthReserveInternode(const NumPhase, PhaseStemElongation,

DryMatStructInternodePop, DryMatStructSheathPop, CoeffResCapacityInternode,

AssimSurplus, ResInternodeMobiliDay: Double;

var ResCapacityInternodePop, IncreaseResInternodePop, DryMatResInternodePop, AssimNotUsed,

AssimNotUsedCum, GrowthResInternodePop : Double);

begin

try

if (NumPhase >= 2) then

begin

ResCapacityInternodePop := (DryMatStructInternodePop + DryMatStructSheathPop) \*

CoeffResCapacityInternode;

IncreaseResInternodePop := Min(Max(AssimSurplus, 0),

Max((ResCapacityInternodePop - DryMatResInternodePop), 0));

GrowthResInternodePop := IncreaseResInternodePop - ResInternodeMobiliDay;

DryMatResInternodePop := DryMatResInternodePop + GrowthResInternodePop;

AssimNotUsed := Max((AssimSurplus - IncreaseResInternodePop), 0);

AssimNotUsedCum := AssimNotUsedCum + AssimNotUsed;

end;

except

AfficheMessageErreur('RS\_EvolGrowthReserveInternode', URisocas);

end;

end;

**Module n°73 - RS\_EvolGrowthTot**

This module calculates total growth of the day.

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - GrowthStructLeafPop** -IN-

**3 - GrowthStructSheathPop** -IN-

**4 - GrowthStructRootPop** -IN-

**5 - GrowthStructInternodePop** -IN-

**6 - GrowthStructPaniclePop** -IN-

**7 - GrowthResInternodePop** -IN-

**8 - PanicleFilPop** -IN-

**9 - GrowthStructTotPop** -OUT-

**10 - GrowthDryMatPop** -OUT-

procedure RS\_EvolGrowthTot(const NumPhase, GrowthStructLeafPop, GrowthStructSheathPop, GrowthStructRootPop, GrowthStructInternodePop, GrowthStructPaniclePop, GrowthResInternodePop, PanicleFilPop : Double; var GrowthStructTotPop, GrowthDryMatPop : Double);

begin

try

if (NumPhase < 5) then

begin

GrowthStructTotPop := GrowthStructLeafPop + GrowthStructSheathPop +

GrowthStructRootPop + GrowthStructInternodePop +

GrowthStructPaniclePop;

end

else

begin

GrowthStructTotPop := 0;

end;

GrowthDryMatPop := GrowthStructTotPop + GrowthResInternodePop + PanicleFilPop;

except

AfficheMessageErreur('RS\_EvolGrowthTot', URisocas);

end;

end;

**Module n°74 - RS\_ExcessAssimilToRoot\_V2**

This module optionally invests daily excess assimilates in root growth (within the limits of potential root wt / soil volume as parameterized), under the condition has choses “**ExcessAssimToRoot = 1”. Otherwise nothing changes. User choice is binary (1 = Yes, 0 = No). Default is No.**

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - ExcessAssimToRoot** -IN-

**3 - DryMatStructRootPop** -IN- (en kg/ha) : Root blade dry matter at population scale

**3 - DryMatStructRootPop** -IN- (en kg/ha) : Root blade dry matter at population scale

**4 - RootSystVolPop** -IN- (en m3)

**5 - CoeffRootMassPerVolMax** -IN- (en kg/m3) : Maximal root dry weight that can be produced per cubic meter of soil explored by root system. Sets demand for root partitioning, resulting value

**6 - RootMassPerVol** -OUT-

**7 - GrowthStructRootPop** -INOUT-

**8 - AssimNotUsed** -INOUT- (en kg/ha/d) : This assimilate is not used because all sinks and the reserve buffer are saturated

procedure RS\_ExcessAssimilToRoot\_V2(const NumPhase, ExcessAssimToRoot, DryMatStructRootPop, RootSystVolPop, CoeffRootMassPerVolMax : Double; var RootMassPerVol, GrowthStructRootPop, AssimNotUsed : Double);

begin

try

if (NumPhase > 1) then

begin

RootMassPerVol := DryMatStructRootPop / RootSystVolPop;

end;

if (ExcessAssimToRoot = 1) then

begin

if (NumPhase < 5) and (NumPhase > 1) and (AssimNotUsed > 0) then

begin

if (RootMassPerVol < CoeffRootMassPerVolMax) then

begin

GrowthStructRootPop := GrowthStructRootPop + AssimNotUsed;

AssimNotUsed := 0;

end;

end;

end;

except

AfficheMessageErreur('RS\_ExcessAssimilToRoot\_V2', URisocas);

end;

end;

**Module n°75 - RS\_EvolDryMatTot\_V2**

This module calculates dry matter of all entities, and also yield components and grain filling status. The latter is the actual grain weight over the potential grain weight given by PanicleSinkPop at flowering. At maturity, the final value of GrainFillingStatus (0…1) permits evaluating whether grain yield was sink limited (=1) or source limited (<1).

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - ChangePhase** -IN- : ce booléen permet de savoir si la journée courante est une journée de changement de phase (facilite l'initialisation)

**3 - PlantsPerHill** -IN- : Number of seeds placed together in a hill (supposing all will germinate and grow)

**4 - TxResGrain** -IN- (en fraction) : Fraction of seed weight mibilizabme for growth of seeding

**5 - PoidsSecGrain** -IN- (en g) : Dry weight of single seed (or filled grain) in g, or 1000-grain dry wt in kg

**6 - Density** -IN- (en pieds/Ha)

**7 - GrowthStructLeafPop** -IN-

**8 - GrowthStructSheathPop** -IN-

**9 - GrowthStructRootPop** -IN-

**10 - GrowthStructInternodePop** -IN-

**11 - GrowthStructPaniclePop** -IN-

**12 - GrowthStructTotPop** -IN-

**13 - GrowthResInternodePop** -IN-

**14 - GrainYieldPop** -IN- (en kg/ha) : Grain yield at population scale (without structural parts of panicle)

**15 - ResCapacityInternodePop** -IN- (en kg/ha) : Size of potential reservoir for reserves in internodes per ha

**16 - CulmsPerPlant** -IN- (en till/plant) : Tiller number per plant (without main stem)

**17 - CoeffPanSinkPop** -IN- (en fraction) : Sets the grain mass (yield) that can be produced per structural mass of panicle including peduncle

**18 - SterilityTot** -IN- (en fraction) : Total spikelet sterility (caused by cold, heat and drought)

**19 - DryMatStructLeafPop** -INOUT- (en kg/ha) : Green leaf blade dry matter at population scale

**20 - DryMatStructSheathPop** -INOUT- (en kg/ha) : Sheath blade dry matter at population scale

**21 - DryMatStructRootPop** -INOUT- (en kg/ha) : Root blade dry matter at population scale

**22 - DryMatStructInternodePop** -INOUT- (en kg/ha) : Internode blade dry matter at population scale (only structural component: reserves are simulated and output separately)

**23 - DryMatStructPaniclePop** -INOUT- (en kg/ha) : Panicle structural dry matter at population scale (does not include grains), formed between PI and flowering

**24 - DryMatStemPop** -INOUT-

**25 - DryMatStructTotPop** -INOUT- (en kg/ha) : Total structural dry matter at population scale (excluding reserves and grains)

**26 - DryMatResInternodePop** -INOUT-

**27 - DryMatVegeTotPop** -OUT- (en kg/ha) : Total vegetative dry matter at population scale (does not include panicles and grains)

**28 - DryMatPanicleTotPop** -OUT- (en kg/ha) : Total panicle dry matter at population scale (includes structural parts and grains)

**29 - DryMatAboveGroundPop** -OUT- (en kg/ha) : Total aboveground dry matter at population scale

**30 - DryMatTotPop** -OUT- (en kg/ha) : Total plant dry matter at population scale including roots

**31 - HarvestIndex** -OUT- (en fraction) : harvest index = grain yield / aboveground dry matter

**32 - InternodeResStatus** -OUT- (en fraction) : Current level of filling of internode reserve reservoir

**33 - PanicleNumPop** -INOUT- (en panicl/ha) : Number of panicles per ha

**34 - PanicleNumPlant** -INOUT- (en panicl/plan) : Number of panicles per plant = number of surviving tillers, considered fertile

**35 - GrainYieldPanicle** -INOUT- (en g/panicl) : grain yield per panicle

**36 - SpikeNumPop** -INOUT- (en spike/ha) : spikelet number per ha (= potential grain number per ha)

**37 - SpikeNumPanicle** -INOUT- (en spike/panic) : spikelet number per panicle (=potential grain number per panicle)

**38 - FertSpikeNumPop** -INOUT- (en spike/ha) : fertile spikelet number per ha (those that are not sterile due to heat, cold or drought)

**39 - GrainFillingStatus** -INOUT- (en g/g) : Degree of realization of filling of fertile spikelets. If <1, this may mean that grain weight is < potential (set by seed weight)

**40 - RootShootRatio** -INOUT- (en fraction) : Dry mass ratio of root over aboveground organs

procedure RS\_EvolDryMatTot\_V2(const NumPhase, ChangePhase, PlantsPerHill, TxResGrain, PoidsSecGrain, Densite, GrowthStructLeafPop, GrowthStructSheathPop, GrowthStructRootPop, GrowthStructInternodePop, GrowthStructPaniclePop, GrowthStructTotPop, GrowthResInternodePop, GrainYieldPop, ResCapacityInternodePop, CulmsPerPlant, CoeffPanSinkPop, SterilityTot : Double; var DryMatStructLeafPop, DryMatStructSheathPop, DryMatStructRootPop, DryMatStructInternodePop, DryMatStructPaniclePop, DryMatStructStemPop, DryMatStructTotPop, DryMatResInternodePop, DryMatVegeTotPop, DryMatPanicleTotPop, DryMatAboveGroundPop, DryMatTotPop, HarvestIndex, InternodeResStatus, PanicleNumPop, PanicleNumPlant, GrainYieldPanicle, SpikeNumPop, SpikeNumPanicle, FertSpikeNumPop, GrainFillingStatus, RootShootRatio : Double);

begin

try

if ((NumPhase = 2) and (ChangePhase = 1)) then

begin

DryMatTotPop := Densite \* PlantsPerHill \* TxResGrain \* PoidsSecGrain / 1000;

DryMatStructLeafPop := DryMatTotPop \* 0.5;

end

else

begin

if (NumPhase > 1) then

begin

DryMatStructLeafPop := DryMatStructLeafPop + GrowthStructLeafPop;

DryMatStructSheathPop := DryMatStructSheathPop + GrowthStructSheathPop;

DryMatStructRootPop := DryMatStructRootPop + GrowthStructRootPop;

DryMatStructInternodePop := DryMatStructInternodePop + GrowthStructInternodePop;

DryMatStructPaniclePop := DryMatStructPaniclePop + GrowthStructPaniclePop;

DryMatStructStemPop := DryMatStructSheathPop + DryMatStructInternodePop +

DryMatResInternodePop;

DryMatStructTotPop := DryMatStructLeafPop + DryMatStructSheathPop +

DryMatStructRootPop + DryMatStructInternodePop +

DryMatStructPaniclePop;

DryMatVegeTotPop := DryMatStructTotPop + DryMatResInternodePop –

DryMatStructPaniclePop;

DryMatPanicleTotPop := DryMatStructPaniclePop + GrainYieldPop;

DryMatTotPop := DryMatVegeTotPop + GrainYieldPop;

DryMatAboveGroundPop := DryMatTotPop - DryMatStructRootPop;

RootShootRatio := DryMatStructRootPop / DryMatAboveGroundPop;

if (ResCapacityInternodePop = 0) then

begin

InternodeResStatus := 0;

end

else

begin

InternodeResStatus := DryMatResInternodePop / ResCapacityInternodePop;

end;

end;

if (NumPhase > 4) then

begin

HarvestIndex := GrainYieldPop / DryMatAboveGroundPop;

PanicleNumPlant := CulmsPerPlant;

PanicleNumPop := CulmsPerPlant \* Densite \* PlantsPerHill;

GrainYieldPanicle := 1000 \* GrainYieldPop / PanicleNumPop;

SpikeNumPop := 1000 \* DryMatStructPaniclePop \* CoeffPanSinkPop / PoidsSecGrain;

SpikeNumPanicle := SpikeNumPop / PanicleNumPop;

FertSpikeNumPop := SpikeNumPop \* (1 - SterilityTot);

GrainFillingStatus := 1000 \* (GrainYieldPop / FertSpikeNumPop) / PoidsSecGrain;

end;

end;

except

AfficheMessageErreur('RS\_EvolDryMatTot\_V2',URisocas);

end;

end;

**Module n°76 - RS\_EvalLai**

This module calculates LAI on the basis of structural leaf dry weight (at population scale, kg/ha) and SLA. (The “correctedSla” variable is a local (module-internal) variable just used to overcome a division by zero problem at beginning of simulation.)

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - ChangePhase** -IN- : ce booléen permet de savoir si la journée courante est une journée de changement de phase (facilite l'initialisation)

**3 - DryMatStructLeafPop** -IN- (en kg/ha) : Green leaf blade dry matter at population scale

**3 - DryMatStructLeafPop** -IN- (en kg/ha) : Green leaf blade dry matter at population scale

**4 - Sla** -IN- (en ha/kg) : Specific leaf area (reciprocal of specific leaf weight). High values indicate thin leaves

**5 - SlaMax** -IN- (en kg/ha) : Initial (maximal) value of SLA (leaf surface/dw) for bulk canopy

**5 - SlaMax** -IN- (en kg/ha) : Initial (maximal) value of SLA (leaf surface/dw) for bulk canopy

**6 - Lai** -OUT- (en m²/m²) : leaf area index (green leaf blades only)

procedure RS\_EvalLai(const NumPhase, ChangePhase, DryMatStructLeafPop, sla, SlaMax : Double; var Lai : Double);

var

CorrectedSla : Double;

begin

try

if ((NumPhase = 2) and (ChangePhase = 1)) then

begin

CorrectedSla := SlaMax;

end

else

begin

CorrectedSla := sla;

end;

Lai := DryMatStructLeafPop \* CorrectedSla;

except

AfficheMessageErreur('RS\_EvalLai', URisocas);

end;

end;

**Module n°77 - RS\_EvalMaximumLai**

This module calculates the maximal green LAI produced by the plant during its cycle, in oder to make the information available for a planned parameter optimization routine.

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - ChangePhase** -IN- : ce booléen permet de savoir si la journée courante est une journée de changement de phase (facilite l'initialisation)

**3 - Lai** -IN- (en m²/m²) : leaf area index (green leaf blades only)

**4 - TempLai** -INOUT- (en m²/m²)

**5 - MaxLai** -INOUT- (en m²/m²) : Valeur maxi du Lai atteinte jusqu'au jour en cours

procedure RS\_EvalMaximumLai(const NumPhase, ChangePhase, Lai : Double; var TempLai, MaximumLai : Double);

begin

try

if (Lai > TempLai) then

begin

TempLai := Lai;

end;

if (NumPhase <> 7) then

begin

MaximumLai := 0;

end

else if (NumPhase = 7) and (ChangePhase = 1) then

begin

MaximumLai := TempLai;

end;

except

AfficheMessageErreur('RS\_EvalMaximumLai', URisocas);

end;

end;

**Module n°78 - RS\_LeafRolling**

This model calculates leaf rolling on the basis of two crop parameters (RollingBase & RollingSens) and environmental variables FTSW (soil drought) and ETo (atmospheric drought). RollingBase sets the fraction of leaf surface that remains exposed to sunlight if the leaf is fully rolled. RollingSens sets the sensitivity of rolling to environment. An intgeractive ETo \* FTSW term is used to calculate Krolling, the coefficient (state variable) expressing the fraction of leaf area exposed to sunlight in its current rolling state. Rolling is totally inactivated if RollingBase is set to 1.

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - RollingBase** -IN- (en fraction) : Leaf rolling under drought: relative leaf blade surface when fully rolled, as fraction of unfolded surface

**3 - RollingSens** -IN- (en none) : Sensitivity of leaf rolling to drought (interactive term of atmospheric drought = PET and FTSW)

**4 - FTSW** -IN- (en none) : fraction of transpirable soil water within the bulk root zone

**5 - ETo** -IN- (en mm/d) : potential evapotranspiration (FAO, also called PET, ETP or Eto). Approximates atmospheric demand for water vapor applied to a calm water surface

**6 - KRolling** -OUT- (en fraction) : current rolling status of leaf rolling due to drought, expressed as fraction of visible rolled surface / potential expanded surface

procedure RS\_LeafRolling(const NumPhase, RollingBase, RollingSens, FTSW, Eto : Double; var KRolling : Double);

begin

try

if (NumPhase > 1) then

begin

KRolling := RollingBase + (1 - RollingBase) \* Power(FTSW, Max(0.0000001, Eto \*

RollingSens));

end;

except

AfficheMessageErreur('RS\_LeafRolling', URisocas);

end;

end;

**Module n°79 - RS\_EvalClumpAndLightInter\_V2**

This module calculates the clumping (heteorgeneity in space) of the leaf canopy, as a function of plant height, width and spacing. Light transmission ratio (Ltr) of the canopy is calculated on the basis of light extinction coefficient (Kdf) without clumping (LTRkdf) or with clumping (LTR kdfcl). Only the latter is used for growth computations.

The previous clumping calculation using a coefficient (crop parameter) was replaced by a simpler one without a specific parameter. The assumption is that the light received by the soil area outside the projection of the plant crown (based on a circle with PlantWidth as diameter ; NOT the projection of leaf area!!!) is ineffective. Consequently, Beer’s law is applied on the basis of the fraction of the soil area that is under the plant crown projection, which leads to a slightly increased local LAI (leaves have more mutual shading),resulting in a slightly reduced light interception per unit field area. The effect is only important under extremely wide spacing or with extremely small plants.

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - KRolling** -IN- (en fraction) : current rolling status of leaf rolling due to drought, expressed as fraction of visible rolled surface / potential expanded surface

**3 - Density** -IN- (en pieds/Ha)

**4 - PlantWidth** -IN- (en mm) : Approximate plant width

**5 - PlantHeight** -IN- (en mm) : Overall height of plant incuding top leaves, assuming vertical orientation

**6 - Kdf** -IN- (en none) : Sets extinction of incoming diffuse solar radiation by crop canopy as function of LAI. Value 0.4 = very erect leaves, 1 = horizontal leaves

**7 - Lai** -IN- (en m²/m²) : leaf area index (green leaf blades only)

**8 - FractionPlantHeightSubmer** -IN- (en mm)

**9 - LIRkdf** -OUT-

**10 - LIRkdfcl** -INOUT- (en fraction) : Light interception rate of canopy as calculated with Kdfcl (taking into account crop Kdf and clumping)

**11 - LTRkdf** -OUT-

**12 - LTRkdfcl** -INOUT- (en fraction) : Light transmission rate of canopy as calculated with Kdfcl (taking into account crop Kdf and clumping), = 1-LIRkdfcl

procedure RS\_EvalClumpAndLightInter\_V2(const NumPhase, KRolling, Density, PlantWidth, PlantHeight, Kdf, Lai, FractionPlantHeightSubmer : Double; var LIRkdf, LIRkdfcl, LTRkdf, LTRkdfcl : Double);

var

RolledLai : Double;

begin

try

if (NumPhase > 1) and (PlantWidth > 0) then

begin

RolledLai := Lai \* KRolling \* Sqrt((1 - FractionPlantHeightSubmer));

LIRkdf := 1 - Exp(-Kdf \* RolledLai);

LIRkdfcl := (1 - Exp(-Kdf \* RolledLai \* 10000 / Min(10000, Density \* pi \*

Power(PlantWidth / 2000, 2)))) \* (Min(10000, Density \* pi \*

Power(PlantWidth / 2000, 2)) / 10000);

LTRkdf := 1 - LIRkdf ;

LTRkdfcl := 1 - LIRkdfcl ;

end;

except

AfficheMessageErreur('RS\_EvalClumpingAndLightInter\_V2', URisocas);

end;

end;

**Module n°80 - RS\_EvalSlaMitch**

This module calculates specific leaf area (SLA). SLA of new leaf drymatter produced in a day is attributed an SLA value according a Mitcherlich function, based on SLAmin, SLAmax and an attenuator AttenMitch. This produces a curvilinearly decreasing function depending on thermal time elapsed. All leaves initially have a maximal SLA (SLAmax). As new leaves are formed that have lower SLA, the overall mean also decreases. This new algorithm avoids forcing a new SLA value onto old leaves, who actually cannot change their SLA any more (weakness in SARRAH model).

Another algorithm implements an effect of low temperatures on the SLA of new leaves (SLAnew). Thus, if daily mean T drops below Topt1, SLA of new leaves decreases (leaves get thicker), attaining SLAmax as T approaches Tbase. This is only effective during early stages of development because towards the end, all leaves attain SLAmax anyway. This mechanism reproduces the commonly observed effect that low temperatures reduce leaf area and make leaves thicker.

**1 - SlaMax** -IN- (en kg/ha) : Initial (maximal) value of SLA (leaf surface/dw) for bulk canopy

**2 - SlaMin** -IN- (en kg/ha) : Final (minimal) value of SLA (leaf surface/dw) for bulk canopy

**3 - AttenMitch** -IN- (en none) : Coefficient for Mitscherlich function leading to non linear evolution of SLA from max to min

**4 - SumDegresDay** -IN- (en °C.jour) : Somme de degrés.jours depuis le début de la phase 1

**5 - SDJLevee** -IN- (en °C.d) : Phase 1. Sets duration from sowing to germination (but may be overrode by drought)

**6 - NumPhase** -IN- (en none) : Phenological phase

**7 - DegresDuJourCor** -IN- (en °C.d) : same, but adjusted for drought effect using a value >0 for DEVcstr: drought slows development, thus reducing the effective heat dose available

**8 - TOpt1** -IN- (en °C) : Lower limit of plateau of Thermal response of development

**9 - TBase** -IN- (en °C) : Base temperature (air based in this model; no microclimate simulated)

**10 - TempSLA** -IN- (en fraction) : Sets sensitivity of SLA of new leaves to non-optimal T

**11 - DryMatStructLeafPop** -IN- (en kg/ha) : Green leaf blade dry matter at population scale

**12 - GrowthStructLeafPop** -IN-

**13 - SlaMitch** -OUT- (en kg/ha)

**14 - SlaNew** -OUT- (en kg/ha)

**15 - Sla** -INOUT- (en ha/kg) : Specific leaf area (reciprocal of specific leaf weight). High values indicate thin leaves

procedure RS\_EvalSlaMitch(const SlaMax, SlaMin, AttenMitch, SDJ, SDJLevee, NumPhase, DegresDuJour, TOpt1, TBase, TempSla, DryMatStructLeafPop, GrowthStructLeafPop : Double; var SlaMitch, SlaNew, Sla : Double);

begin

try

if (NumPhase > 1) then

begin

SlaMitch := SlaMin + (SlaMax - SlaMin) \* Power(AttenMitch, (SDJ - SDJLevee));

SlaNew := SlaMin + (SlaMitch - SlaMin) \* Power(DegresDuJour / (TOpt1 - TBase), TempSla);

Sla := ((Sla \* DryMatStructLeafPop) + (SlaNew \* GrowthStructLeafPop)) /

(DryMatStructLeafPop + GrowthStructLeafPop);

end

else

begin

SlaMitch := 0;

SlaNew := 0;

Sla := SlaMax;

end;

except

AfficheMessageErreur('RS\_EvalSlaMitch', URisocas);

end;

end;

**Module n°81 - RS\_EvalRuiss\_FloodDyna\_V2**

This module implements, after a rain or irrigation event, the runoff, filling of macropores and floodwater compartment, and water management interventions (surface drainage). By sub-module:

1. implement lifesaving drainage:

If this option is chosen (parameter LifeSavingDrainage set to 1) the surface floodwater will be drained to the depth of ½ plant height whenever floodwaterdepth is greater than this limit. The drained water is considered as runoff (Lr).

1. implement terminal drainage

The user can choose a surface drainage date (in days after flowering) after which BundHeight will be considered zero. Drained surface water will be considered as runoff (Lr).

1. implement runoff and EauDispo under terminal drainage

Implementation of runoff (Lr) and calculation of available free surface and soil water in a situation of terminal drainage

1. implement classical upland runoff (SARRAH)

Implementation of runoff under upland conditions as set by soil parameters (IRD model). **Note:** If deep drainage (Dr) in upland situation exceeds PercolationMax (soil parameter), the excess is added to runaoff (Lr) as calculated in module **RS\_EvolWaterLoggingUpland\_V2.**

1. implement bunded-plot style water ponding and runoff, regular situation w/o drainage

Regular calculation of runoff (Lr; considered as spill-over here) in a bunded situation.

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - Pluie** -IN- (en mm) : Pluviométrie journalière

**3 - SeuilRuiss** -IN- (en mm) : Seuil pluie, calcul du ruissellement (cf PourcRuiss)

**4 - PourcRuiss** -IN- (en %) : Pourcentage de ruissellement de la quantité de pluie supérieure au seuil de ruissellement

**5 - BundHeight** -IN- (en mm) : Bunds leading to surface floodwater storage. No lateral seepage is simulated

**6 - Irrigation** -IN- (en mm) : Quantité nette d'eau apportée par irrigation (tenir compte de l'efficience)

**7 - PlantHeight** -IN- (en mm) : Overall height of plant incuding top leaves, assuming vertical orientation

**8 - LifeSavingDrainage** -IN- (en fraction) : If value=1 then plots are automatically drained down to 50% of plant height in order to avoid submergence

**9 - PlotDrainageDAF** -IN- : Performs automatic plot surface drainage at X DAF (days after flowering). If value 99 is chosen, no drainage happens

**10 - VolMacropores** -IN-

**11 - SeuilRuiss** -IN- (en mm) : Seuil pluie, calcul du ruissellement (cf PourcRuiss)

**12 - PercolationMax** -IN- (en mm) : Percolation (deep drainage) daily rate in bunded plots if standing water and/or macropores filled with water

**13 - DAF** -IN- (en d)

**14 - StockMacropores** -INOUT-

**15 - FloodwaterDepth** -INOUT- (en mm)

**16 - EauDispo** -INOUT- (en mm) : Total available water column stored in soil profile

**17 - Lr** -INOUT- (en mm/d) : Runoff

procedure RS\_EvalRuiss\_FloodDyna\_V2(const NumPhase, Rain, SeuilRuiss, PourcRuiss, BundHeight, Irrigation, PlantHeight, LifeSavingDrainage, PlotDrainageDAF, VolMacropores, SuilRuiss, PercolationMax, DAF : Double; var StockMacropores, FloodwaterDepth, EauDispo, Lr : Double);

var

CorrectedIrrigation : Double;

CorrectedBundheight : Double;

Begin

try

Lr := 0;

CorrectedBundheight := Bundheight;

// implement lifesaving drainage (1)

if (LifeSavingDrainage = 1) and

(FloodwaterDepth > (0.5 \* PlantHeight)) and

(PlantHeight > 0) and

(NumPhase > 1) and

(BundHeight > 0) then

begin

CorrectedBundheight := 0.5 \* PlantHeight;

Lr := Lr + Max(0, FloodwaterDepth - (0.5 \* PlantHeight));

FloodwaterDepth := Min(FloodwaterDepth, (0.5 \* PlantHeight));

if (FloodwaterDepth + StockMacropores > 0) then

begin

EauDispo := FloodwaterDepth + StockMacropores;

end;

end;

// implement terminal drainage (2)

if (NumPhase > 4) and (NumPhase < 7) and (DAF > PlotDrainageDAF) and (BundHeight > 0) then

begin

CorrectedBundHeight := 0;

Lr := Lr + FloodwaterDepth;

FloodWaterDepth := 0;

if ((FloodwaterDepth + StockMacropores) > 0) then

begin

EauDispo := StockMacropores;

end

else

begin

EauDispo := Rain;

end;

end;

// define corrected irrigation

if (Irrigation = NullValue) then

begin

CorrectedIrrigation := 0;

end

else

begin

CorrectedIrrigation := Irrigation;

end;

// implement runoff and EauDispo under terminal drainage (3)

if (CorrectedBundHeight = 0) and (BundHeight <> CorrectedBundHeight) then

begin

if ((StockMacropores + FloodwaterDepth) = 0) then

begin

EauDispo := Rain + CorrectedIrrigation;

end

else

begin

StockMacropores := StockMacropores + Rain + CorrectedIrrigation;

Lr := Lr + Max(0, StockMacropores - VolMacropores);

StockMacropores := StockMacropores - Max(0, StockMacropores - VolMacropores);

EauDispo := StockMacropores;

end;

end;

// implement classical upland runoff (SARRAH) (4)

if (BundHeight = 0) then

begin

if (Rain > SuilRuiss) then

begin

Lr := Lr + (Rain + CorrectedIrrigation - SeuilRuiss) \* PourcRuiss / 100;

EauDispo := Rain + CorrectedIrrigation - Lr;

end

else

begin

EauDispo := Rain + CorrectedIrrigation;

end;

end;

// implement bunded-plot style water ponding and runoff, regular situation w/o drainage(5)

if (CorrectedBundHeight > 0) then

begin

if ((StockMacropores + FloodwaterDepth) = 0) then

begin

Lr := Lr + Max((Rain + CorrectedIrrigation - BundHeight - VolMacropores), 0);

EauDispo := Min(Rain + CorrectedIrrigation, BundHeight + VolMacropores);

end

else

begin

StockMacropores := StockMacropores + Rain + CorrectedIrrigation;

FloodwaterDepth := FloodwaterDepth + Max(0, StockMacropores - VolMacropores);

StockMacropores := Min(VolMacropores, StockMacropores);

Lr := Lr + Max(0, FloodwaterDepth - CorrectedBundHeight);

FloodwaterDepth := Min(FloodwaterDepth, CorrectedBundHeight);

EauDispo := StockMacropores + FloodwaterDepth;

end;

end;

except

AfficheMessageErreur('RS\_EvalRuiss\_FloodDyna\_V2', URisocas);

end;

end;

**Module n°82 - RS\_AutomaticIrrigation\_V2**

This module calculates the automatic irrigation process (option under bunded lowland conditions when BundHeight is set to >0). The field is irrigated daily, as needed, to achieve either floodwaterDepth >= Bundheight (para) \* IrrigAutoTarget (para), or half of plant height, what ever is smaller. This way, irrigation does not submerge young plants. If the option is chosen to drain the plots somewhere between flowering and maturity (parameter PlotDrainageDaf), automatic irrigation is stopped.

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - IrrigAuto** -IN- (en none) : If value=1 then daily automatic irrigation is performed to either a fraction of BundHeight (parameter IriigAutoTarget) of 50% of plant height

**3 - IrrigAutoTarget** -IN- (en fraction) : Fraction of BundHeight to be achieved with automatic irrigation. E.g., if value=0.8 then water will be introduced up to 80% of BundHeight

**4 - BundHeight** -IN- (en mm) : Bunds leading to surface floodwater storage. No lateral seepage is simulated

**5 - PlantHeight** -IN- (en mm) : Overall height of plant incuding top leaves, assuming vertical orientation

**6 - Irrigation** -IN- (en mm) : Quantité nette d'eau apportée par irrigation (tenir compte de l'efficience)

**7 - PlotDrainageDAF** -IN- : Performs automatic plot surface drainage at X DAF (days after flowering). If value 99 is chosen, no drainage happens

**8 - DAF** -IN- (en d)

**9 - VolMacropores** -IN-

**10 - VolRelMacropores** -IN- (en %) : Rel. Volume of macropores in soil (%) = air spaces that are filled with air when soil saturated but freely drained

**11 - Pluie** -IN- (en mm) : Pluviométrie journalière

**12 - FloodwaterDepth** -INOUT- (en mm)

**13 - IrrigAutoDay** -OUT- (en mm)

**14 - IrrigTotDay** -OUT- (en mm)

**15 - StockMacropores** -INOUT-

**16 - EauDispo** -INOUT- (en mm) : Total available water column stored in soil profile

procedure RS\_AutomaticIrrigation\_V2(const NumPhase, IrrigAuto, IrrigAutoTarget, BundHeight, PlantHeight, Irrigation, PlotDrainageDAF, DAF, VolMacropores, VolRelMacropores, Rain : Double; var FloodwaterDepth, IrrigAutoDay, IrrigTotDay, StockMacropores, EauDispo: Double);

var

IrrigAutoTargetCor : Double;

CorrectedIrrigation : Double;

CorrectedBundHeight : Double;

begin

try

CorrectedBundHeight := BundHeight;

if (Irrigation = NullValue) then

begin

CorrectedIrrigation := 0;

end

else

begin

CorrectedIrrigation := Irrigation;

end;

if (NumPhase > 4) and (NumPhase < 7) and (DAF > PlotDrainageDAF) then

begin

CorrectedBundHeight := 0;

end;

if (NumPhase < 7) and (DAF <= PlotDrainageDaf) and (IrrigAuto = 1) and (NumPhase > 0) and (CorrectedBundHeight > 0) then

begin

IrrigAutoTargetCor := Min((IrrigAutoTarget \* BundHeight), (0.5 \* PlantHeight));

// Provide initial water flush for infiltration

if (NumPhase = 1) then

begin

IrrigAutoTargetCor := Max(IrrigAutoTargetCor, BundHeight / 2);

end;

// dimension irrigation on day i

IrrigAutoDay := Max(0, (IrrigAutoTargetCor - FloodwaterDepth + Min((VolMacropores - StockMacropores) / 2, VolRelMacropores \* 200 / 100)));

if (StockMacropores + FloodwaterDepth) = 0 then

begin

EauDispo := Rain + CorrectedIrrigation + IrrigAutoDay;

end

else

begin

FloodwaterDepth := FloodwaterDepth + IrrigAutoDay;

// make sure Macropores is fully filled before floodwater can build up!

if (VolMacropores > 0) and (StockMacropores < VolMacropores) and (FloodwaterDepth > 0) then

begin

StockMacropores := StockMacropores + FloodwaterDepth ;

FloodwaterDepth := max(0, StockMacropores - VolMacropores);

StockMacropores := StockMacropores - FloodwaterDepth;

end;

EauDispo := StockMacropores + FloodwaterDepth;

end;

end

else

begin

if (NumPhase < 7) and (DAF <= PlotDrainageDaf) and (IrrigAuto = 1) and (NumPhase > 0) and (CorrectedBundHeight = 0) then

begin

FloodwaterDepth := 0;

StockMacropores := 0;

end;

end;

IrrigTotDay := CorrectedIrrigation + IrrigAutoDay;

except

AfficheMessageErreur('RS\_AutomaticIrrigation\_V2', URisocas);

end;

end;

**Module n°83 - RS\_EvolRempliResRFE\_RDE\_V2**

This module calculates replenishment of soil, macropores and floodwater as new water has come into the system through rain or irrigation. Either an upland situation (BundHeight=0) or and bunded situation (BundHeight>0; rainfed or irrigated lowland) is considered. Water that cannot be stored is considered deep drainage (Dr).

A detailed description of processes will follow…

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - RuSurf** -IN- (en mm) : Reserve utile de l'horizon de surface

**3 - EauDispo** -IN- (en mm) : Total available water column stored in soil profile

**4 - RuRac** -IN- (en mm) : Water column that can potentially be strored in soil volume explored by root system

**5 - CapaRFE** -IN- (en mm) : Capacité du réservoir facilement évaporable (au potentiel)

**6 - CapaREvap** -IN- (en mm) : Capacité du réservoir d'évaporation

**7 - CapaRDE** -IN- (en mm) : Réserve difficilement transpirable mais évaporable

**8 - StRuMax** -IN- (en mm) : Capacité maximale de la RU

**9 - PercolationMax** -IN- (en mm) : Percolation (deep drainage) daily rate in bunded plots if standing water and/or macropores filled with water

**10 - BundHeight** -IN- (en mm) : Bunds leading to surface floodwater storage. No lateral seepage is simulated

**11 - EpaisseurSurf** -IN- (en mm) : Epaisseur de l'horizon de surface

**12 - EpaisseurProf** -IN- (en mm) : Epaisseur de l'horizon de profondeur

**13 - VolMacropores** -IN-

**14 - FloodwaterDepth** -INOUT- (en mm)

**15 - StockTotal** -INOUT- (en mm) : Total water column stored in soil profile

**16 - StockRac** -INOUT- (en mm) : Water column stored in soil volume explored by root system

**17 - Hum** -INOUT- (en mm) : Quantité d'eau maximum jusqu'au front d'humectation

**18 - StockSurface** -INOUT- (en mm) : Water column stored in topsoil layer

**19 - Dr** -OUT- (en mm/d) : Deep drainage

**20 - ValRDE** -INOUT- (en mm) : Contenu de la RDE

**21 - ValRFE** -INOUT- (en mm) : Contenu de la RFE

**22 - ValRSurf** -INOUT- (en mm) : Contenu des 2 réservoirs RDE et REvap

**23 - FloodwaterGain** -OUT- (en mm)

**24 - StockMacropores** -INOUT-

procedure RS\_EvolRempliResRFE\_RDE\_V2(const NumPhase, RuSurf, EauDispo, RuRac, CapaRFE, CapaREvap, CapaRDE, StRuMax, PercolationMax, BundHeight, EpaisseurSurf, EpaisseurProf, VolMacropores : Double; var FloodwaterDepth, StockTotal, StockRac, Hum, StockSurface, Dr, ValRDE, ValRFE, ValRSurf, FloodwaterGain, StockMacropores : Double);

var

EauReste, ValRSurfPrec, EauTranspi : Double;

begin

try

Dr := 0;

EauTranspi := 0;

if (StockMacropores + FloodwaterDepth = 0) then

begin

EauReste := 0;

ValRFE := ValRFE + EauDispo;

if (ValRFE > CapaRFE) then

begin

EauReste := ValRFE - CapaRFE;

ValRFE := CapaRFE;

end;

ValRSurfPrec := ValRSurf;

ValRSurf := ValRSurf + EauReste;

if (ValRSurfPrec < CapaREvap) then

begin

EauTranspi := EauDispo - (Min(CapaREvap, ValRSurf) - ValRSurfPrec);

end

else

begin

EauTranspi := EauDispo;

end;

if (ValRSurf > (CapaREvap + CapaRDE)) then

begin

ValRSurf := CapaREvap + CapaRDE;

ValRDE := CapaRDE;

end

else

begin

if (ValRSurf <= CapaREvap) then

begin

ValRDE := 0;

end

else

begin

ValRDE := ValRSurf - CapaREvap;

end;

end;

StockSurface := ValRFE + ValRDE;

StockTotal := StockTotal + EauTranspi;

If (StockTotal > StRuMax) then

begin

Dr := StockTotal - StRuMax;

StockTotal := StRuMax;

end

else

begin

Dr := 0;

end;

if Hum < (CapaRFE + CapaRDE) then

begin

Hum := StockSurface;

end

else

begin

Hum := Max(Hum, StockTotal);

end;

end;

StockRac := Min(StockRac + EauTranspi, RuRac);

// Shifting non-percolating Dr back to macropores & floodwater if plot is bunded

if (BundHeight > 0) then

begin

// Shifting non-percolating Dr to Floodwater

StockMacropores := StockMacropores + Max(0, Dr - PercolationMax);

Dr := Min(Dr, PercolationMax);

if (StockMacropores > VolMacropores) then

begin

FloodWaterDepth := FloodWaterDepth + (StockMacropores - VolMacropores);

StockMacropores := VolMacropores;

end;

// Implementing Dr

if (FloodwaterDepth >= PercolationMax) then

begin

Dr := PercolationMax ;

FloodwaterDepth := FloodwaterDepth - Dr ;

StockMacropores := VolMacropores ;

end

else

begin

if (FloodwaterDepth < PercolationMax) and ((FloodwaterDepth + StockMacropores) >= PercolationMax) then

begin

Dr := PercolationMax ;

FloodwaterDepth := FloodwaterDepth - Dr ;

StockMacropores := StockMacropores + FloodwaterDepth ;

FloodwaterDepth := 0 ;

end

else

begin

Dr := Min(PercolationMax, (FloodwaterDepth + StockMacropores + Dr));

FloodwaterDepth := 0 ;

StockMacropores := 0 ;

end;

end;

end;

except

AfficheMessageErreur('RS\_EvolRempliResRFE\_RDE\_V2', URisocas);

end;

end;

**Module n°84 - RS\_EvolWaterLoggingUpland\_V2**

This module implements for an upland situation (BundHeight=0) an upper limit to soil deep drainage (PercoltionMax), permitting the simulation of soil water logging. The amount of deep drainage (Dr) that cannot percolate builds up in the macropopres (air spaces of the soil), from bottom to top. If the macropores are full, the excess is added to runoff (Lr). Soil water logging can be simulated as a stress depending on the type of crop (see subsequent module).

**1 - PercolationMax** -IN- (en mm) : Percolation (deep drainage) daily rate in bunded plots if standing water and/or macropores filled with water

**2 - BundHeight** -IN- (en mm) : Bunds leading to surface floodwater storage. No lateral seepage is simulated

**3 - VolMacropores** -IN-

**4 - Dr** -INOUT- (en mm/d) : Deep drainage

**5 - Lr** -INOUT- (en mm/d) : Runoff

**6 - StockMacropores** -INOUT-

procedure RS\_EvolWaterLoggingUpland\_V2(const PercolationMax, BundHeight, VolMacropores : Double; var Dr, Lr, StockMacropores : Double);

begin

try

if (Dr > PercolationMax) and (BundHeight = 0) then

begin

StockMacropores := StockMacropores + (Dr - PercolationMax);

Lr := Lr + Max(0, StockMacropores - VolMacropores);

Dr := PercolationMax;

StockMacropores := Min(StockMacropores, VolMacropores);

end;

except

AfficheMessageErreur('RS\_EvolWaterLoggingUpland\_V2', URisocas);

end;

end;

**Module n°85 - RS\_EvalStressWaterLogging\_V2**

This module calculates the fraction (0…1) of the root system (in terms of root depth) that is water logged. On this basis, using a crop sensitivity coefficient set by user, the stress coefficient **CoeffStressLogging** is calculated that is used elsewhere to reduce transpiration and photosynthesis (because water logging closes stomata in sensitives genotypes).

**1 - StockMacropores** -IN-

**2 - VolMacropores** -IN-

**3 - RootFront** -IN- (en mm) : depth of root front

**4 - EpaisseurSurf** -IN- (en mm) : Epaisseur de l'horizon de surface

**5 - EpaisseurProf** -IN- (en mm) : Epaisseur de l'horizon de profondeur

**6 - WaterLoggingSens** -IN- (en none)

**7 - FractionRootsLogged** -OUT- (en none)

**8 - CoeffStressLogging** -OUT- (en none)

procedure RS\_EvalStressWaterLogging\_V2(const StockMacropores, VolMacropores, RootFront, EpaisseurSurf, EpaisseurProf, WaterLoggingSens : Double; var FractionRootsLogged, CoeffStressLogging : Double);

begin

try

if (StockMacropores > 0) and (RootFront > 0) then

begin

FractionRootsLogged := (Max(0, RootFront - ((VolMacropores - StockMacropores) /

VolMacropores) \* (EpaisseurSurf + EpaisseurProf))) / RootFront;

CoeffStressLogging := 1 - (FractionRootsLogged \* Min(1, WaterLoggingSens));

end;

except

AfficheMessageErreur('RS\_EvalStressWaterLogging\_V2',URisocas);

end;

end;

**Module n°86 - RS\_EvolRempliMacropores\_V2**

This module just updates soil water state variables after the soil water movements calculated in previous modules.(The module name is badly chosen.)

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - EpaisseurSurf** -IN- (en mm) : Epaisseur de l'horizon de surface

**3 - EpaisseurProf** -IN- (en mm) : Epaisseur de l'horizon de profondeur

**4 - ResUtil** -IN- (en mm/m)

**5 - StockMacropores** -IN-

**6 - RootFront** -IN- (en mm) : depth of root front

**7 - CapaRDE** -IN- (en mm) : Réserve difficilement transpirable mais évaporable

**8 - CapaRFE** -IN- (en mm) : Capacité du réservoir facilement évaporable (au potentiel)

**9 - FloodwaterDepth** -IN- (en mm)

**10 - StockTotal** -INOUT- (en mm) : Total water column stored in soil profile

**11 - Hum** -INOUT- (en mm) : Quantité d'eau maximum jusqu'au front d'humectation

**12 - StockSurface** -INOUT- (en mm) : Water column stored in topsoil layer

**13 - StockRac** -INOUT- (en mm) : Water column stored in soil volume explored by root system

**14 - ValRDE** -INOUT- (en mm) : Contenu de la RDE

**15 - ValRFE** -INOUT- (en mm) : Contenu de la RFE

**16 - ValRSurf** -INOUT- (en mm) : Contenu des 2 réservoirs RDE et REvap

procedure RS\_EvolRempliMacropores\_V2(const NumPhase, EpaisseurSurf, EpaisseurProf, RU, StockMacropores, RootFront, CapaRDE, CapaRFE, FloodwaterDepth : Double; var StockTotal, Hum, StockSurface, StockRac, ValRDE, ValRFE, ValRSurf : Double);

begin

try

if ((StockMacropores + FloodwaterDepth) > 0) then

begin

StockTotal := (EpaisseurSurf + EpaisseurProf) \* RU / 1000 + StockMacropores;

Hum := StockTotal;

StockSurface := EpaisseurSurf \* RU / 1000 + (EpaisseurSurf / (EpaisseurSurf +

EpaisseurProf)) \* StockMacropores;

StockRac := RootFront \* RU / 1000 + (RootFront / (EpaisseurSurf + EpaisseurProf)) \*

StockMacropores;

ValRDE := CapaRDE;

ValRFE := CapaRFE;

ValRSurf := EpaisseurSurf \* RU / 1000;

end;

except

AfficheMessageErreur('RS\_EvolRempliMacropores\_V2', URisocas);

end;

end;

**Module n°87 - RS\_EvolRurRFE\_RDE\_V2**

This module calculates the changing access to soil water as the root front proceeds deeper into the soil. The compartments of available water (potential: RuRac; actual: StockRac) are updated.

**1 - VitesseRacinaire** -IN- (en mm/jour) : Vitesse racinaire journalière

**2 - Hum** -IN- (en mm) : Quantité d'eau maximum jusqu'au front d'humectation

**3 - ResUtil** -IN- (en mm/m)

**4 - StockSurface** -IN- (en mm) : Water column stored in topsoil layer

**5 - RuSurf** -IN- (en mm) : Reserve utile de l'horizon de surface

**6 - ProfRacIni** -IN- (en mm) : Profondeur de semis ou profondeur initiale des racines simulation en cours du cycle

**7 - EpaisseurSurf** -IN- (en mm) : Epaisseur de l'horizon de surface

**8 - EpaisseurProf** -IN- (en mm) : Epaisseur de l'horizon de profondeur

**9 - ValRDE** -IN- (en mm) : Contenu de la RDE

**10 - ValRFE** -IN- (en mm) : Contenu de la RFE

**11 - NumPhase** -IN- (en none) : Phenological phase

**12 - ChangePhase** -IN- : ce booléen permet de savoir si la journée courante est une journée de changement de phase (facilite l'initialisation)

**13 - FloodwaterDepth** -IN- (en mm)

**14 - StockMacropores** -IN-

**15 - RuRac** -INOUT- (en mm) : Water column that can potentially be strored in soil volume explored by root system

**16 - StockRac** -INOUT- (en mm) : Water column stored in soil volume explored by root system

**17 - StockTotal** -INOUT- (en mm) : Total water column stored in soil profile

**18 - FloodwaterGain** -INOUT- (en mm)

**19 - RootFront** -INOUT- (en mm) : depth of root front

procedure RS\_EvolRurRFE\_RDE\_V2(const VitesseRacinaire, Hum, Ru, StockSurface, RuSurf, ProfRacIni, EpaisseurSurf, EpaisseurProf, ValRDE, ValRFE, NumPhase, ChangePhase, FloodwaterDepth, StockMacropores : Double; var RuRac, StockRac, StockTotal, FloodwaterGain, RootFront : Double);

var

DeltaRur : Double;

begin

try

if (NumPhase = 0) then

begin

RuRac := 0;

StockRac := 0;

end

else

begin

if ((NumPhase = 1) and (ChangePhase = 1)) then // les conditions de germination sont atteinte et nous sommes le jour même

begin

RuRac := Ru \* Min(ProfRacIni, (EpaisseurSurf + EpaisseurProf)) / 1000;

if (ProfRacIni <= EpaisseurSurf) then

begin

StockRac := (ValRDE + ValRFE) \* ProfRacIni / EpaisseurSurf;

end

else

begin

StockRac := StockTotal \* Min(ProfRacIni / (EpaisseurSurf + EpaisseurProf), 1);

end;

end

else

begin

if (Hum - StockMacropores - RuRac) < (VitesseRacinaire / 1000 \* RU) then

begin

DeltaRur := Max(0, Hum - StockMacropores - RuRac);

end

else

begin

DeltaRur := VitesseRacinaire / 1000 \* RU;

end;

RuRac := RuRac + DeltaRur;

if (RuRac > RuSurf) then

begin

StockRac := StockRac + DeltaRur;

end

else

begin

StockRac := (ValRDE + ValRFE) \* (RuRac / RuSurf);

end;

end;

end;

// The following is needed to have the correct basis for calculating FTSW under

// supersaturated soil condition (macropores filled)

if (NumPhase <> 0) then

begin

RootFront := ((1 / Ru) \* RuRac) \* 1000;

end;

if ((StockMacropores + FloodwaterDepth) > 0) then

begin

StockRac := RootFront \* Ru / 1000 + (RootFront / (EpaisseurSurf + EpaisseurProf)) \* StockMacropores;

StockRac := Min(StockRac, StockTotal);

end;

except

AfficheMessageErreur('RS\_EvolRurRFE\_RDE\_V2', URisocas);

end;

end;

**Module n°88 - RS\_PlantSubmergence\_V2**

This module calculates the fraction of plant height submerged by floodwater under bunded conditions. This is needed to calculate the reduction of photosynthesis caused by this.

**1 - PlantHeight** -IN- (en mm) : Overall height of plant incuding top leaves, assuming vertical orientation

**2 - FloodwaterDepth** -IN- (en mm)

**3 - FractionPlantHeightSubmer** -OUT- (en mm)

procedure RS\_PlantSubmergence\_V2(const PlantHeight, FloodwaterDepth : Double; var FractionPlantHeightSubmer : Double);

begin

try

FractionPlantHeightSubmer := Min(Max(0, FloodwaterDepth / Max(PlantHeight, 0.1)), 1);

except

AfficheMessageErreur('RS\_PlantSubmergence\_V2', URisocas);

end;

end;

**Module n°89 - RS\_EvalRootFront**

This module calculates the current depth of the root front in mm. In fact, the progression of the root front is calculated in soil potential water storage accesses (RuRac; in mm water column) . This is converted here into absolute depth.

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - RuRac** -IN- (en mm) : Water column that can potentially be strored in soil volume explored by root system

**3 - ResUtil** -IN- (en mm/m)

**4 - RootFront** -OUT- (en mm) : depth of root front

procedure RS\_EvalRootFront(const NumPhase, RuRac, Ru : Double; var RootFront : Double);

begin

try

if (NumPhase > 0) then

begin

RootFront := ((1 / Ru) \* RuRac) \* 1000;

end;

except

AfficheMessageErreur('RS\_EvalRootFront', URisocas);

end;

end;

**Module n°90 - RS\_EvolPSPMVMD**

This module calculates a component of the Vaksmann-Dingkuhn « Impatience » model of photoperiodism. Explanation follows… (not yet done)

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - ChangePhase** -IN- : ce booléen permet de savoir si la journée courante est une journée de changement de phase (facilite l'initialisation)

**3 - SumDegreDayCor** -IN- (en °C.jour)

**4 - DegresDuJourCor** -IN- (en °C.d) : same, but adjusted for drought effect using a value >0 for DEVcstr: drought slows development, thus reducing the effective heat dose available

**7 - DayLength** -IN- (en hour(dec)) : day length including civil twilight

**8 - PPExp** -IN- (en none) : Attenuator for progressive PSP response to PP. Rarely used in calibration procedure, a robust value is 0.17

**10 - SumDDPhasePrec** -INOUT- (en °C.jour) : Somme en degrés/jour de la phase précédente

**11 - SeuilTemp** -INOUT- (en °C.jour) : Seuil des températures cumulées pour la phase en cours

procedure RS\_EvolPSPMVMD(const Numphase, ChangePhase, SomDegresJourCor, DegresDuJourCor, SeuilPP, PPCrit, DureeDuJour, PPExp : Double; var SumPP, SeuilTempPhasePrec, SeuilTempPhaseSuivante : Double);

var

SDJPSP : Double;

begin

try

if (NumPhase = 3) then

begin

if (ChangePhase = 1) then

begin

SumPP := 100; //valeur arbitraire d'initialisation >2.5

SDJPSP := Max(0.01, DegresDuJourCor);

end

else

begin

SDJPSP := SomDegresJourCor - SeuilTempPhasePrec + Max(0.01, DegresDuJourCor);

end;

SumPP := Power((1000 / SDJPSP), PPExp) \* Max(0, (DureeDuJour - PPCrit)) / (SeuilPP - PPCrit);

SeuilTempPhaseSuivante := SeuilTempPhasePrec + SDJPSP;

end;

except

AfficheMessageErreur('RS\_EvolPSPMVMD', URisocas);

end;

end;

**Module n°91 - EvolSomDegresJour**

This module cumulates daily heat units (degree-days) during crop development.

Cumul des degres jour.

**1 - DegresDuJour** -IN- (en °C.d) : daily heat dose (in degree-days)

**2 - NumPhase** -IN- (en none) : Phenological phase

**3 - SumDegresDay** -INOUT- (en °C.jour) : Somme de degrés.jours depuis le début de la phase 1

procedure RS\_EvolSomDegresJour(const DegresDuJour, NumPhase : Double; var SommeDegresJour : Double);

begin

try

if (NumPhase >= 1) then // on ne cumule qu'après la germination

begin

SommeDegresJour := SommeDegresJour + DegresDuJour;

end

else

begin

SommeDegresJour := 0;

end;

except

AfficheMessageErreur('RS\_EvolSomDegresJour', URisocas);

end;

end;

**Module n°92 - RS\_EvolSomDegresJourCor**

This module cumulates the variable SommeDegresJourCor, which is the daily number of heat units corrected for drought effects, based on the crop parameter DevCstr. If it is set to zero, SommeDegresJourCor eaquals SommeDegresJour. If it is 1 or intermediate, the daily heat units are reduced under drought, thus slowing down development.

**1 - DegresDuJourCor** -IN- (en °C.d) : same, but adjusted for drought effect using a value >0 for DEVcstr: drought slows development, thus reducing the effective heat dose available

**2 - NumPhase** -IN- (en none) : Phenological phase

**3 - SumDegreDayCor** -INOUT- (en °C.jour)

procedure RS\_EvolSomDegresJourCor(const DegresDuJourCor, NumPhase : Double; var SommeDegresJourCor : Double);

begin

try

if (NumPhase >= 1) then // on ne cumule qu'après la germination

begin

SommeDegresJourCor := SommeDegresJourCor + DegresDuJourCor;

end

else

begin

SommeDegresJourCor := 0;

end;

except

AfficheMessageErreur('RS\_EvolSomDegresJourCor', URisocas);

end;

end;

**Module n°93 - RS\_EvalRUE**

This module simulates ecological and crop balance variables for output. Balance variables include cumulative expressions of resources used or produced (CumXXX) and efficiency expressions (ratios of two cumulatice resources or products). The variables are calculated throughout the crop simulation and thus represent at any point in time the cumulative status of all past fluxes. Cumulated entities are: Tr, ET (Tr+Evap), Irrigation, Drainage (Dr), Runoff (Lr), total water received, total water used. Efficiency variables are: effective radiation use efficiency (RUE), instantaneaous TE (TrEffInst; non cumulative), TE (TrEff), WUE based on ET (WueEt) and WUE based on total water used (WueTot). Daily effective conversion efficiency is also calculated (ConversionEff), which includes effects of drought, chilling, submergence, transplanting shock and water logging.

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - ChangePhase** -IN- : ce booléen permet de savoir si la journée courante est une journée de changement de phase (facilite l'initialisation)

**3 - PARIntercepte** -IN- (en MJ/m²/d) : PAR intercepted by crop

**4 - DryMatTotPop** -IN- (en kg/ha) : Total plant dry matter at population scale including roots

**4 - DryMatTotPop** -IN- (en kg/ha) : Total plant dry matter at population scale including roots

**5 - DeadLeafdrywtPop** -IN- (en kg/ha) : Dead leaf dry mass (assuming they do not decompose; but exluding the mass that has been recycled)

**5 - DeadLeafdrywtPop** -IN- (en kg/ha) : Dead leaf dry mass (assuming they do not decompose; but exluding the mass that has been recycled)

**6 - Tr** -IN- (en mm/d) : Actual crop transpiration

**7 - Evap** -IN- (en mm/d) : Actual soil surface evaporation under crop (if any is present)

**8 - Dr** -IN- (en mm/d) : Deep drainage

**9 - Lr** -IN- (en mm/d) : Runoff

**10 - SupplyTot** -IN- (en kg/ha/d) : Net fresh assimilate supply per day = Assim-RespMaintTot

**11 - AssimNotUsed** -IN- (en kg/ha/d) : This assimilate is not used because all sinks and the reserve buffer are saturated

**12 - Irrigation** -IN- (en mm) : Quantité nette d'eau apportée par irrigation (tenir compte de l'efficience)

**13 - IrrigAutoDay** -IN- (en mm)

**14 - Pluie** -IN- (en mm) : Pluviométrie journalière

**15 - Assim** -IN- (en kg/ha/d) : Assim=AssimPot \* Cstr (if applicable, corrected with CstrAssim)

**16 - AssimPot** -IN- (en kg/ha/d) : Canopu CH20 assimilation per day BEFORE reduction by stomatal closure (mediated by Cstr) and subtraction of Rm

**17 - Conversion** -IN- (en kg/ha/MJ)

**18 - NbJAS** -IN- (en d) : days after sowing

**19 - RUE** -OUT- (en g/MJ) : radiation use efficiency as calculated frim simulated aboveground dry matter and cumulative PAR intercepted

**21 - CumTr** -INOUT-

**22 - CumEt** -INOUT-

**23 - CumWUse** -INOUT-

**24 - CumWReceived** -INOUT-

**25 - CumIrrig** -INOUT-

**26 - CumDr** -INOUT-

**27 - CumLr** -INOUT-

**28 - TrEffInst** -OUT- (en kg/kg) : Instantaneous Transpiration Efficiency

**29 - TrEff** -OUT- (en kg/kg) : Accrued Transpiration Efficiency

**30 - WueEt** -OUT- (en kg/kg) : Accrued water use efficiency on evapotranspiration basis

**31 - WueTot** -OUT- (en kg/kg) : Accrued water use efficiency on total water use basis including runoff and drainage

**32 - ConversionEff** -OUT- (en g/MJ) : Final conversion of intercepted PAR into assimilation BEFORE respiration

procedure RS\_EvalRUE(const NumPhase, ChangePhase, ParIntercepte, DryMatTotPop, DeadLeafDrywtPop, Tr, Evap, Dr, Lr, SupplyTot, AssimNotUsed, Irrigation, IrrigAutoDay, Pluie, Assim, AssimPot, Conversion, NbJas : Double; var RUE, CumPar, CumTr, CumEt, CumWUse, CumWReceived, CumIrrig, CumDr, CumLr, TrEffInst, TrEff, WueEt, WueTot, ConversionEff : Double);

var

CorrectedIrrigation : Double;

begin

try

if ((NumPhase < 1) or ((NumPhase = 1) and (ChangePhase = 1))) then

begin

CumPar := 0;

RUE := 0;

CumTr := 0.00001;

CumEt := 0.00001;

CumWUse := 0.00001;

CumWReceived := 0;

CumIrrig := 0;

CumDr := 0;

CumLr := 0;

end

else

begin

CumPar := CumPar + ParIntercepte;

CumTr := CumTr + Tr;

CumEt := CumEt + Tr + Evap;

CumWUse := CumWUse + Tr + Evap + Dr + Lr;

if (Irrigation = NullValue) then

begin

CorrectedIrrigation := 0;

end

else

begin

CorrectedIrrigation := Irrigation;

end;

CumWReceived := CumWReceived + Pluie + CorrectedIrrigation + IrrigAutoDay;

CumIrrig := CumIrrig + CorrectedIrrigation + IrrigAutoDay;

CumDr := CumDr + Dr;

CumLr := CumLr + Lr;

if (AssimPot <> 0) then

begin

ConversionEff := Conversion \* Assim / AssimPot ;

end;

if ((Tr > 0) and (NbJas > 15) and (NumPhase > 1)) then

begin

TrEffInst := (SupplyTot - AssimNotUsed) / (Tr \* 10000);

TrEff := DryMatTotPop / (CumTr \* 10000);

WueEt := DryMatTotPop / (CumEt \* 10000);

WueTot := DryMatTotPop / (CumWuse \* 10000);

RUE := ((DryMatTotPop + DeadLeafDrywtPop) / Max(CumPar, 0.00001)) / 10;

end;

end;

except

AfficheMessageErreur('RS\_EvalRUE', URisocas);

end;

end;

**Module n°94 - SorghumMortality**

This module declares the crop dead and ends the simulation if anywhere between germination and maturity the floating mean of the drought stress coefficient over 5 consecutive days is smaller than the crop parameter SeuilStressMortality. This parameter value should ne near zero (0.0001…0.1). If it is set to zero, mortality is not simulated. If the crop dies due to this mechanism, the simulation jumps to NumPhase 7 (end of crop cycle).

**1 - Cstr** -IN- (en none) : drought stress coefficient: FTSW is transformed into Cstr by FAO function using P-factor

**2 - SeuilCstrMortality** -IN- (en d) : Sets the cumulative, uninterrupted drought necessary to kill the plant (simulation ends)

**3 - NumPhase** -INOUT- (en none) : Phenological phase

procedure SorghumMortality(const cstr, SeuilCstrMortality : Double; var NumPhase : double) ;

var

i : Integer;

MoyenneCstr : Double;

begin

try

if (NumPhase >= 2) then

begin

NbJourCompte := NbJourCompte + 1;

// gestion de l'indice...

if (tabCstrIndiceCourant = 5) then

begin

tabCstrIndiceCourant := 1;

tabCstr[tabCstrIndiceCourant] := cstr;

end

else

begin

tabCstrIndiceCourant := tabCstrIndiceCourant + 1;

tabCstr[tabCstrIndiceCourant] := cstr;

end;

// gestion de la mortalité

if (NbJourCompte >= 5) then

begin // on peut moyenner...

MoyenneCstr := 0;

for i := 1 to 5 do

begin

MoyenneCstr := MoyenneCstr + tabCstr[i];

end;

if ((MoyenneCstr / 5) <= SeuilCstrMortality) then

begin

NumPhase:=7;

end;

end;

end;

except

AfficheMessageErreur('SorghumMortality',URiz);

end;

end;

**Module n°95 - RS\_KeyResults\_V2**

This module calculates key outputs of the simulation (final grain yield, biomass, reserves, culm number; maximal LAI and culm number; phase means of Cstr, FTSW and Ic…) for numerical output (no graphics).

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - CulmsPerPlant** -IN- (en till/plant) : Tiller number per plant (without main stem)

**3 - CulmsPerHill** -IN-

**4 - Cstr** -IN- (en none) : drought stress coefficient: FTSW is transformed into Cstr by FAO function using P-factor

**5 - FTSW** -IN- (en none) : fraction of transpirable soil water within the bulk root zone

**6 - Ic** -IN- (en g/g) : state variable "index of competition" = daily assimilate supply/demand

**7 - Lai** -IN- (en m²/m²) : leaf area index (green leaf blades only)

**8 - GrainYieldPop** -IN- (en kg/ha) : Grain yield at population scale (without structural parts of panicle)

**9 - DryMatAboveGroundPop** -IN- (en kg/ha) : Total aboveground dry matter at population scale

**10 - DryMatResInternodePop** -IN-

**11 - CulmsPerPlantMax** -INOUT-

**12 - CulmsPerHillMax** -INOUT-

**13 - DurPhase1** -INOUT-

**14 - DurPhase2** -INOUT-

**15 - DurPhase3** -INOUT-

**16 - DurPhase4** -INOUT-

**17 - DurPhase5** -INOUT-

**18 - DurPhase6** -INOUT-

**19 - CumCstrPhase2** -INOUT-

**20 - CumCstrPhase3** -INOUT-

**21 - CumCstrPhase4** -INOUT-

**22 - CumCstrPhase5** -INOUT-

**23 - CumCstrPhase6** -INOUT-

**24 - CumFTSWPhase2** -INOUT-

**25 - CumFTSWPhase3** -INOUT-

**26 - CumFTSWPhase4** -INOUT-

**27 - CumFTSWPhase5** -INOUT-

**28 - CumFTSWPhase6** -INOUT-

**29 - CumIcPhase2** -INOUT-

**30 - CumIcPhase3** -INOUT-

**31 - CumIcPhase4** -INOUT-

**32 - CumIcPhase5** -INOUT-

**33 - CumIcPhase6** -INOUT-

**34 - IcPhase2** -INOUT-

**35 - IcPhase3** -INOUT-

**36 - IcPhase4** -INOUT-

**37 - IcPhase5** -INOUT-

**38 - IcPhase6** -INOUT-

**39 - FtswPhase2** -INOUT-

**40 - FtswPhase3** -INOUT-

**41 - FtswPhase4** -INOUT-

**42 - FtswPhase5** -INOUT-

**43 - FtswPhase6** -INOUT-

**44 - CstrPhase2** -INOUT-

**45 - CstrPhase3** -INOUT-

**46 - CstrPhase4** -INOUT-

**47 - CstrPhase5** -INOUT-

**48 - CstrPhase6** -INOUT-

**49 - DurGermFlow** -INOUT-

**50 - DurGermMat** -INOUT-

**51 - LaiFin** -INOUT-

**52 - CulmsPerHillFin** -INOUT-

**53 - CulmsPerPlantFin** -INOUT-

**54 - GrainYieldPopFin** -INOUT-

**55 - DryMatAboveGroundPopFin** -INOUT-

**56 - ReservePopFin** -INOUT-

procedure RS\_KeyResults\_V2(const NumPhase, CulmsPerPlant, CulmsPerHill, Cstr, FTSW, Ic, Lai, GrainYieldPop, DryMatAboveGroundPop, DryMatResInternodePop : Double; var CulmsPerPlantMax, CulmsPerHillMax, DurPhase1, DurPhase2, DurPhase3, DurPhase4, DurPhase5, DurPhase6, CumCstrPhase2, CumCstrPhase3, CumCstrPhase4, CumCstrPhase5, CumCstrPhase6, CumFTSWPhase2, CumFTSWPhase3, CumFTSWPhase4, CumFTSWPhase5, CumFTSWPhase6, CumIcPhase2, CumIcPhase3, CumIcPhase4, CumIcPhase5, CumIcPhase6, IcPhase2, IcPhase3, IcPhase4, IcPhase5, IcPhase6, FtswPhase2, FtswPhase3, FtswPhase4, FtswPhase5, FtswPhase6, CstrPhase2, CstrPhase3, CstrPhase4, CstrPhase5, CstrPhase6, DurGermFlow, DurGermMat, LaiFin, CulmsPerHillFin, CulmsPerPlantFin, GrainYieldPopFin, DryMatAboveGroundPopFin, ReservePopFin : Double);

begin

try

if (NumPhase > 1) and (NumPhase < 7) then

begin

CulmsPerPlantMax := Max(CulmsPerPlant, CulmsPerPlantMax);

CulmsPerHillMax := Max(CulmsPerHill, CulmsPerHillMax);

end;

if (NumPhase = 1) then

begin

DurPhase1 := DurPhase1 + 1;

end;

if (NumPhase = 2) then

begin

DurPhase2 := DurPhase2 + 1;

CumCstrPhase2 := CumCstrPhase2 + Cstr;

CumFTSWPhase2 := CumFTSWPhase2 + FTSW;

CumIcPhase2 := CumIcPhase2 + Ic;

end;

if (NumPhase = 3) then

begin

DurPhase3 := DurPhase3 + 1;

CumCstrPhase3 := CumCstrPhase3 + Cstr;

CumFTSWPhase3 := CumFTSWPhase3 + FTSW;

CumIcPhase3 := CumIcPhase3 + Ic;

end;

if (NumPhase = 4) then

begin

DurPhase4 := DurPhase4 + 1;

CumCstrPhase4 := CumCstrPhase4 + Cstr;

CumFTSWPhase4 := CumFTSWPhase4 + FTSW;

CumIcPhase4 := CumIcPhase4 + Ic;

end;

if (NumPhase = 5) then

begin

DurPhase5 := DurPhase5 + 1;

CumCstrPhase5 := CumCstrPhase5 + Cstr;

CumFTSWPhase5 := CumFTSWPhase5 + FTSW;

CumIcPhase5 := CumIcPhase5 + Ic;

end;

if (NumPhase = 6) then

begin

DurPhase6 := DurPhase6 + 1;

CumCstrPhase6 := CumCstrPhase6 + Cstr;

CumFTSWPhase6 := CumFTSWPhase6 + FTSW;

CumIcPhase6 := CumIcPhase6 + Ic;

end;

if (NumPhase = 7) then

begin

IcPhase2 := CumIcPhase2 / Max(DurPhase2, 0.1);

IcPhase3 := CumIcPhase3 / Max(DurPhase3, 0.1);

IcPhase4 := CumIcPhase4 / Max(DurPhase4, 0.1);

IcPhase5 := CumIcPhase5 / Max(DurPhase5, 0.1);

IcPhase6 := CumIcPhase6 / Max(DurPhase6, 0.1);

FtswPhase2 := CumFtswPhase2 / Max(DurPhase2, 0.1);

FtswPhase3 := CumFtswPhase3 / Max(DurPhase3, 0.1);

FtswPhase4 := CumFtswPhase4 / Max(DurPhase4, 0.1);

FtswPhase5 := CumFtswPhase5 / Max(DurPhase5, 0.1);

FtswPhase6 := CumFtswPhase6 / Max(DurPhase6, 0.1);

CstrPhase2 := CumCstrPhase2 / Max(DurPhase2, 0.1);

CstrPhase3 := CumCstrPhase3 / Max(DurPhase3, 0.1);

CstrPhase4 := CumCstrPhase4 / Max(DurPhase4, 0.1);

CstrPhase5 := CumCstrPhase5 / Max(DurPhase5, 0.1);

CstrPhase6 := CumCstrPhase6 / Max(DurPhase6, 0.1);

DurGermFlow := DurPhase2 + DurPhase3 + DurPhase4;

DurGermMat := DurPhase2 + DurPhase3 + DurPhase4 + DurPhase5 + DurPhase6;

LaiFin := Lai;

CulmsPerHillFin := CulmsPerHill;

CulmsPerPlantFin := CulmsPerPlant;

GrainYieldPopFin := GrainYieldPop;

DryMatAboveGroundPopFin := DryMatAboveGroundPop;

ReservePopFin := DryMatResInternodePop;

end;

except

AfficheMessageErreur('RS\_KeyResults\_V2', URisocas);

end;

end;

**Module n°96 - RS\_ResetVariablesToZero**

This module resets crop state variables to zero after crop maturity.

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - ChangePhase** -IN- : ce booléen permet de savoir si la journée courante est une journée de changement de phase (facilite l'initialisation)

**3 - CulmsPerPlant** -INOUT- (en till/plant) : Tiller number per plant (without main stem)

**4 - CulmsPerHill** -INOUT-

**5 - CulmsPop** -INOUT- (en till/ha) : Tiller number per ha (without main stem)

**6 - GrainYieldPop** -INOUT- (en kg/ha) : Grain yield at population scale (without structural parts of panicle)

**7 - DryMatStructLeafPop** -INOUT- (en kg/ha) : Green leaf blade dry matter at population scale

**8 - DryMatStructSheathPop** -INOUT- (en kg/ha) : Sheath blade dry matter at population scale

**9 - DryMatStructRootPop** -INOUT- (en kg/ha) : Root blade dry matter at population scale

**10 - DryMatStructInternodePop** -INOUT- (en kg/ha) : Internode blade dry matter at population scale (only structural component: reserves are simulated and output separately)

**11 - DryMatResInternodePop** -INOUT-

**12 - DryMatStructPaniclePop** -INOUT- (en kg/ha) : Panicle structural dry matter at population scale (does not include grains), formed between PI and flowering

**13 - DryMatStemPop** -INOUT-

**14 - DryMatStructTotPop** -INOUT- (en kg/ha) : Total structural dry matter at population scale (excluding reserves and grains)

**15 - DryMatVegeTotPop** -INOUT- (en kg/ha) : Total vegetative dry matter at population scale (does not include panicles and grains)

**16 - DryMatPanicleTotPop** -INOUT- (en kg/ha) : Total panicle dry matter at population scale (includes structural parts and grains)

**17 - DryMatAboveGroundPop** -INOUT- (en kg/ha) : Total aboveground dry matter at population scale

**18 - DryMatTotPop** -INOUT- (en kg/ha) : Total plant dry matter at population scale including roots

**19 - HarvestIndex** -INOUT- (en fraction) : harvest index = grain yield / aboveground dry matter

**20 - PanicleNumPop** -INOUT- (en panicl/ha) : Number of panicles per ha

**21 - PanicleNumPlant** -INOUT- (en panicl/plan) : Number of panicles per plant = number of surviving tillers, considered fertile

**22 - GrainYieldPanicle** -INOUT- (en g/panicl) : grain yield per panicle

**23 - SpikeNumPop** -INOUT- (en spike/ha) : spikelet number per ha (= potential grain number per ha)

**23 - SpikeNumPop** -INOUT- (en spike/ha) : spikelet number per ha (= potential grain number per ha)

**24 - SpikeNumPanicle** -INOUT- (en spike/panic) : spikelet number per panicle (=potential grain number per panicle)

**25 - FertSpikeNumPop** -INOUT- (en spike/ha) : fertile spikelet number per ha (those that are not sterile due to heat, cold or drought)

**26 - GrainFillingStatus** -INOUT- (en g/g) : Degree of realization of filling of fertile spikelets. If <1, this may mean that grain weight is < potential (set by seed weight)

**27 - PhaseStemElongation** -INOUT- (en none) : Indicates whether internodes are elongating (1) or not (0)

**28 - Sla** -INOUT- (en ha/kg) : Specific leaf area (reciprocal of specific leaf weight). High values indicate thin leaves

**29 - HaunIndex** -INOUT- (en none) : Number of leaves appeared on main stem, including those that have already senesced

**30 - ApexHeight** -INOUT- (en mm) : Height of growing point over ground (excluding the panicle and its peduncle)

**31 - PlantHeight** -INOUT- (en mm) : Overall height of plant incuding top leaves, assuming vertical orientation

**32 - PlantWidth** -INOUT- (en mm) : Approximate plant width

**33 - VitesseRacinaireDay** -INOUT- (en mm/d) : current progression rate of root front

**34 - Kcl** -INOUT- (en none) : coefficient of clumping

**35 - KRolling** -INOUT- (en fraction) : current rolling status of leaf rolling due to drought, expressed as fraction of visible rolled surface / potential expanded surface

**36 - LIRkdfcl** -INOUT- (en fraction) : Light interception rate of canopy as calculated with Kdfcl (taking into account crop Kdf and clumping)

**37 - LTRkdfcl** -INOUT- (en fraction) : Light transmission rate of canopy as calculated with Kdfcl (taking into account crop Kdf and clumping), = 1-LIRkdfcl

**38 - AssimPot** -INOUT- (en kg/ha/d) : Canopu CH20 assimilation per day BEFORE reduction by stomatal closure (mediated by Cstr) and subtraction of Rm

**39 - Assim** -INOUT- (en kg/ha/d) : Assim=AssimPot \* Cstr (if applicable, corrected with CstrAssim)

**40 - RespMaintTot** -INOUT- (en kg/ha/d) : Total daily maintenance respiration (Rm), sum of that of all organs as calculated with organ specific coefficients

**41 - SupplyTot** -INOUT- (en kg/ha/d) : Net fresh assimilate supply per day = Assim-RespMaintTot

**42 - AssimSurplus** -INOUT- (en kg/ha/d) : Daily assimilate surplus after allocation to structural growth and grain filling. This surplus goes into internode storage

**43 - AssimNotUsed** -INOUT- (en kg/ha/d) : This assimilate is not used because all sinks and the reserve buffer are saturated

**44 - AssimNotUsedCum** -INOUT- (en kg/ha) : Accrued term of AssimNotUsed

**45 - TillerDeathPop** -INOUT- (en tiller/d/ha) : Daily number of senesced tillers per ha

**46 - DeadLeafdrywtPop** -INOUT- (en kg/ha) : Dead leaf dry mass (assuming they do not decompose; but exluding the mass that has been recycled)

**47 - ResCapacityInternodePop** -INOUT- (en kg/ha) : Size of potential reservoir for reserves in internodes per ha

**48 - InternodeResStatus** -INOUT- (en fraction) : Current level of filling of internode reserve reservoir

**49 - Cstr** -INOUT- (en none) : drought stress coefficient: FTSW is transformed into Cstr by FAO function using P-factor

**50 - FTSW** -INOUT- (en none) : fraction of transpirable soil water within the bulk root zone

procedure RS\_ResetVariablesToZero(const NumPhase, ChangePhase : Double; varCulmsPerPlant, CulmsPerHill, CulmsPop, GrainYieldPop, DryMatStructLeafPop, DryMatStructSheathPop, DryMatStructRootPop, DryMatStructInternodePop, DryMatResInternodePop, DryMatStructPaniclePop, DryMatStructStemPop, DryMatStructTotPop, DryMatVegeTotPop, DryMatPanicleTotPop, DryMatAboveGroundPop, DryMatTotPop, HarvestIndex, PanicleNumPop, PanicleNumPlant, GrainYieldPanicle, SpikeNumPop, SpikeNumPanicle, FertSpikePop, GrainFillingStatus, PhaseStemElongation, Sla, HaunIndex, ApexHeight, PlantHeight, PlantWidth, VitesseRacinaireDay, Kcl, KRolling, LIRKdfcl, LtrKdfcl, AssimPot, Assim, RespMaintTot, SupplyTot, AssimSurplus, AssimNotUsed, AssimNotUsedCum, TillerDeathPop, DeadLeafDryWtPop, ResCapacityInternodePop, InternodeResStatus, cstr, FTSW : Double);

begin

try

if ((NumPhase = 7) and (ChangePhase = 1)) then

begin

CulmsPerPlant := 0;

CulmsPerHill := 0;

CulmsPop := 0;

GrainYieldPop := 0;

DryMatStructLeafPop := 0;

DryMatStructSheathPop := 0;

DryMatStructRootPop := 0;

DryMatStructInternodePop := 0;

DryMatResInternodePop := 0;

DryMatStructPaniclePop := 0;

DryMatStructStemPop := 0;

DryMatStructTotPop := 0;

DryMatVegeTotPop := 0;

DryMatPanicleTotPop := 0;

DryMatAboveGroundPop := 0;

DryMatTotPop := 0;

HarvestIndex := 0;

PanicleNumPop := 0;

PanicleNumPlant := 0;

GrainYieldPanicle := 0;

SpikeNumPop := 0;

SpikeNumPanicle := 0;

FertSpikePop := 0;

GrainFillingStatus := 0;

PhaseStemElongation := 0;

Sla := 0;

HaunIndex := 0;

ApexHeight := 0;

PlantHeight := 0;

PlantWidth := 0;

VitesseRacinaireDay := 0;

Kcl := 0;

KRolling := 0;

LIRKdfcl := 0;

LTRKdfcl := 1;

AssimPot := 0;

Assim := 0;

RespMaintTot := 0;

SupplyTot := 0;

AssimSurplus := 0;

AssimNotUsed := 0;

AssimNotUsedCum := 0;

TillerDeathPop := 0;

DeadLeafDryWtPop := 0;

ResCapacityInternodePop := 0;

InternodeResStatus := 0;

cstr := 0;

FTSW := 0;

end;

except

AfficheMessageErreur('RS\_ResetVariablesToZero', URisocas);

end;

end;

**Module n°97 - RS\_EvalSimEndCycle**

This module ends the crop cycle and determines the total crop duration in days.

**1 - NumPhase** -IN- (en none) : Phenological phase

**2 - ChangePhase** -IN- : ce booléen permet de savoir si la journée courante est une journée de changement de phase (facilite l'initialisation)

**3 - NbJAS** -IN- (en d) : days after sowing

**4 - SimEndCycle** -INOUT- (en d)

procedure RS\_EvalSimEndCycle(const NumPhase, ChangePhase, NbJas : Double; var SimEndCycle : Double);

begin

try

if (NumPhase = 7) and (ChangePhase = 1) then

begin

SimEndCycle := NbJas

end;

except

AfficheMessageErreur('RS\_EvalSimEndCycle', URisocas);

end;

end;