**Modèle SarrahV32**

(Documentation générée le )

**Module n°1 - BhyInitPlot** Procédure Delphi: InitPlot, s'exécute à l'initialisation du modèle :

**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 - Ru** -IN- (en mm/m) : Réserve utile par mètre de sol

**4 - ProfRacIni** -IN- (en mm) : Profondeur de semis ou profondeur initiale des racines simulation en cours du cycle

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

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

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

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

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

**10 - Ltr** -OUT- (en MJ/MJ) : Taux de rayonnement transmis au sol

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

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

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

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

procedure InitPlot(const StockIniSurf, StockIniProf, Ru, ProfRacIni, EpaisseurSurf, EpaisseurProf, DateSemis : double; var StSurf, StTot, Ltr, Hum, StRurMax, RuSurf, ProfRU : double);

begin

try

StSurf := StockIniSurf;

Ltr := 1;

StRurMax := Ru \* ProfRacIni / 1000;

RuSurf := EpaisseurSurf / 1000 \* Ru;

StTot := StockIniSurf + StockIniProf;

ProfRU := EpaisseurSurf + EpaisseurProf;

Hum := max(StTot, StRurMax);

except

AfficheMessageErreur('InitPlot',UBilEau);

end;

end;

**Module n°2 - InitiationCulture** Procédure Delphi: InitiationCulture, s'exécute à l'initialisation du modèle :

Initialisation des variables culture

**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 - BiomasseAerienne** -OUT- (en kg/ha) : Biomasse aérienne

**8 - BiomasseAerienne** -OUT- (en kg/ha) : Biomasse aérienne

**9 - BiomasseVegetative** -OUT- (en kg/ha) : Biomasse végétative (racine, feuilles, tige)

**9 - BiomasseVegetative** -OUT- (en kg/ha) : Biomasse végétative (racine, feuilles, tige)

**10 - BiomasseTotale** -OUT- (en kg/ha) : Biomasse totale

**10 - BiomasseTotale** -OUT- (en kg/ha) : Biomasse totale

**11 - BiomasseTiges** -OUT- (en kg/ha) : Biomasse de la tige

**11 - BiomasseTiges** -OUT- (en kg/ha) : Biomasse de la tige

**12 - BiomasseRacinaire** -OUT- (en kg/ha) : Biomasse des racines

**12 - BiomasseRacinaire** -OUT- (en kg/ha) : Biomasse des racines

**13 - BiomasseFeuilles** -OUT- (en kg/ha) : Biomasse des feuilles

**13 - BiomasseFeuilles** -OUT- (en kg/ha) : Biomasse des feuilles

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

**15 - DBiomTot** -OUT- (en kg/ha) : Biomasse totale créée dans la journée

**15 - DBiomTot** -OUT- (en kg/ha) : Biomasse totale créée dans la journée

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

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

procedure InitiationCulture(const SeuilTempLevee, SeuilTempBVP, SeuilTempRPR, SeuilTempMatu1, SeuilTempMatu2 : double; var SommeDegresJourMaximale, NumPhase, BiomasseAerienne, BiomasseVegetative, BiomasseTotale, BiomasseTiges, BiomasseRacinaire, BiomasseFeuilles, SommeDegresJour, DeltaBiomasseTotale, SeuilTempPhaseSuivante, Lai : Double);

begin

try

NumPhase := 0;

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

SommeDegresJour := 0;

BiomasseAerienne := 0;

BiomasseVegetative := 0;

BiomasseTotale := 0;

BiomasseTiges := 0;

BiomasseRacinaire := 0;

BiomasseFeuilles := 0;

DeltaBiomasseTotale := 0;

SeuilTempPhaseSuivante:=0;

Lai := 0;

except

AfficheMessageErreur('InitiationCulture',UMilBilanCarbone);

end;

end;

**Module n°3 - Meteo0DegToRad** Procédure Delphi: DegToRad, s'exécute à l'initialisation du modèle :

**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°4 - Meteo1AVGTempHum** Procédure Delphi: AVGTempHum, s'exécute tous les jours sur les entités concernées :

**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°5 - Meteo2Decli** Procédure Delphi: EvalDecli, s'exécute tous les jours sur les entités concernées :

**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°6 - Meteo3SunPosi** Procédure Delphi: EvalSunPosi, s'exécute tous les jours sur les entités concernées :

**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°7 - Meteo4DayLength** Procédure Delphi: EvalDayLength, s'exécute tous les jours sur les entités concernées :

**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°8 - Meteo5SunDistance** Procédure Delphi: EvalSunDistance, s'exécute tous les jours sur les entités concernées :

**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°9 - Meteo6RayExtra** Procédure Delphi: EvalRayExtra, s'exécute tous les jours sur les entités concernées :

**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°10 - Meteo7RgMax** Procédure Delphi: EvalRgMax, s'exécute tous les jours sur les entités concernées :

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

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

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°11 - Meteo8InsToRg** Procédure Delphi: InsToRg, s'exécute tous les jours sur les entités concernées :

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

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

**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°12 - Meteo9Par** Procédure Delphi: EvalPar, s'exécute tous les jours sur les entités concernées :

**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°13 - MeteoEToFAO** Procédure Delphi: EToFao, s'exécute tous les jours sur les entités concernées :

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

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

**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°14 - EvolPhenoSarrahV3** Procédure Delphi: EvolPhenoSarrahV3, s'exécute tous les jours sur les entités concernées :

Gestion des 7 stades pheno pour des céréales

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

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

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

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

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

**6 - 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

**7 - SeuilEauSemis** -IN- (en mm) : Nombre de mm d'eau efficace pour déclancher le semis.

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

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

**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

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

**13 - PhasePhotoper** -INOUT- (en Numerique) : Egal a 1 pendant la phase photper, 0 sinon

procedure EvolPhenoSarrahV3(const SommeDegresJour, SeuilTempLevee, SeuilTempBVP, SeuilTempRPR, SeuilTempMatu1, SeuilTempMatu2, SeuilPluie, StockSurface : Double; var NumPhase, SommeDegresJourPhasePrec, SeuilTempPhaseSuivante, ChangePhase, PhasePhotoper : Double);

{Méthode générique pour le test de fin de la phase photopériodique.

PhasePhotoper = 0 en fin de la phase photoper et = 1 en debut de la phase

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 somme de degré jours 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 les céréales:

// 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 (seulement pour le mais et riz) Pas pris en compte ici!

// 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 stade grain laiteux

// 6 : du début du stade grain laiteux au jour de récolte

// 7 : le jour de la récolte

}

var

ChangementDePhase, ChangementDeSousPhase : Boolean;

begin

try

ChangePhase := 0;

if (Trunc(NumPhase) = 0) then

begin

if (StockSurface >= SeuilPluie) then

begin

NumPhase := 1;

ChangePhase := 1;

SeuilTempPhaseSuivante := SeuilTempLevee;

end;

end

else

begin

if ((Trunc(NumPhase) = 2) and (SommeDegresJour >= SeuilTempPhaseSuivante)) then

begin

ChangementDePhase := True

end

else

begin

If (Trunc(NumPhase) <> 3) Then

begin

ChangementDePhase := (SommeDegresJour >= SeuilTempPhaseSuivante)

end

else

begin

ChangementDePhase := (PhasePhotoper = 0);

end;

end;

if ChangementDePhase then

begin

ChangePhase := 1;

NumPhase := NumPhase + 1;

SommeDegresJourPhasePrec := SeuilTempPhaseSuivante;

case Trunc(NumPhase) of

1 : SeuilTempPhaseSuivante:= SeuilTempLevee;

2 : SeuilTempPhaseSuivante := SeuilTempPhaseSuivante + SeuilTempBVP;

3 : PhasePhotoper := 1;

4 : SeuilTempPhaseSuivante := SommeDegresJour + SeuilTempRPR;

5 : SeuilTempPhaseSuivante := SeuilTempPhaseSuivante + SeuilTempMatu1;

6 : SeuilTempPhaseSuivante := SeuilTempPhaseSuivante + SeuilTempMatu2;

end;

end;

end;

except

AfficheMessageErreur('EvolPhenoSarrahV3 | NumPhase: '+FloatToStr(NumPhase)+

' SommeDegresJour: '+FloatToStr(SommeDegresJour)+

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

end;

end;

**Module n°15 - EvalDegresJourVitMoy** Procédure Delphi: EvalDegresJourVitMoy, s'exécute tous les jours sur les entités concernées :

calcule les DJ à partir de la linéarisation des courbes de vitesse de développement

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

procedure EvalDegresJourVitMoy(const TMax, TMin, TBase, TOpt1, TOpt2, TL : Double; var DegresDuJour : Double);

var

v, v1, v2 : Double;

begin

try

v1 := ((max(TMin, TBase) + Min(TOpt1, min(TL, TMax)))/2 - TBase) / (TOpt1 - TBase);

v2 := (TL - (max(TMax, TOpt2) + TOpt2) / 2) / (TL - TOpt2);

v := (v1 \* (min(min(TL, TMax), TOpt1) - TMin) + (min(TOpt2, max(TOpt1, min(TL, TMax))) - TOpt1) + v2 \* (max(TOpt2, min(TL, TMax)) - TOpt2)) / (min(TL, TMax)- max(TMin, TBase));

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

except

AfficheMessageErreur('EvalDegresJourVitMoy | TMax='+FloatToStr(TMax)+

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

' TOpt2='+FloatToStr(TOpt2)+' TL='+FloatToStr(TL)+' DegresDuJour='+FloatToStr(DegresDuJour),UMilBilanCarbone);

end;

end;

**Module n°16 - EvolSomDegresJour** Procédure Delphi: EvolSomDegresJour, s'exécute tous les jours sur les entités concernées :

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 EvolSomDegresJour(const DegresDuJour, NumPhase : Double; var SommeDegresJour : Double);

begin

try

if (NumPhase >= 1) then

begin

SommeDegresJour := SommeDegresJour + DegresDuJour;

end

else

begin

SommeDegresJour := 0;

end;

except

AfficheMessageErreur('EvolSommeDegresJour | DegresDuJour: '+FloatToStr(DegresDuJour)+

' Phase n°'+FloatToStr(NumPhase)+

' SommeDegresJour: '+FloatToStr(SommeDegresJour)+

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

end;

end;

**Module n°17 - BhyRunOff** Procédure Delphi: EvalRunOffScale, s'exécute tous les jours sur les entités concernées :

**1 - Pluie** -IN- (en mm) : Pluviométrie journalière

**2 - Irrigation** -IN- (en mm) : Quantité nette d'eau apportée par irrigation (tenir compte de l'efficience)

**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 - EauDispo** -OUT- (en mm) : Total available water column stored in soil profile

**6 - Lr** -OUT- (en mm/d) : Runoff

procedure EvalRunOffScale (const Rain, Irrig, seuilRuiss, PourcRuiss : Double; var EauDispo, Lr : Double);

var

EnPlus : Double;

begin

try

Lr := 0;

if (Irrig = nullValue) then

begin

Enplus := Rain;

end

else

begin

Enplus := Rain + Irrig;

end;

If (Enplus > SeuilRuiss) then

begin

Lr := (Enplus - SeuilRuiss) \* PourcRuiss / 100;

end;

EauDispo := Enplus - Lr;

except

AfficheMessageErreur('EvalRunOffScale',UBilEau);

end;

end;

**Module n°18 - EvolRurCstr** Procédure Delphi: EvolRurCstr, s'exécute tous les jours sur les entités concernées :

Vitesse racinaire contrainte par le stress en eau

**1 - VitesseRacinaire** -IN- (en mm/jour) : Vitesse racinaire journalière

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

**3 - Ru** -IN- (en mm/m) : Réserve utile par mètre de sol

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

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

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

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

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

procedure EvolRurCstr(const Vrac, Hum, Ru, stRuSurf, RuSurf, cstr : Double; var stRurMax, stRur : Double);

{

La vitesse d'enracinement potentielle de la plante peut etre bloque

par manque d'eau en profondeur (Hum). La profondeur d'humectation

est convertie en quantite d'eau maximum equivalente

// Parametres

IN:

Vrac : mm

Hum : mm

StRuSurf : mm

RU : mm/m

RuSurf : mm/m

INOUT:

stRurMax : mm

stRur : mm

}

var

DeltaRur, DayVrac : Double;

begin

try

// on prend une partie si blocage ou toute la nouvelle prof

// avec une quantite d'eau au max.

DayVrac := (VRac \* min(Cstr + 0.3, 1)) / 1000 \* RU;

if ((Hum - StRurMax) < DayVrac) then

begin

DeltaRur := Hum - StRurMax;

end

else

begin

DeltaRur := DayVrac;

end;

StRurMax := StRurMax + Deltarur;

// si les racines vont au dela du stock de surface

// on ajoute le deltaRUR

// sinon on prend au prorata de la profondeur et du stock de surface

if (StRurMax > RuSurf) then

begin

StRur := StRur + Deltarur;

end

else

begin

StRur := max((StRuSurf - Rusurf / 10) \* (StRurMax / RuSurf), 0);

end;

except

AfficheMessageErreur('EvolRur | RuSurf: '+FloatToStr(RuSurf),UBilEau);

end;

end;

**Module n°19 - BhyFilingWater** Procédure Delphi: RempliRes, s'exécute tous les jours sur les entités concernées :

**1 - Ru** -IN- (en mm/m) : Réserve utile par mètre de sol

**2 - ProfRu** -IN- (en mm) : Profondeur maximale de sol

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

**4 - EauDispo** -IN- (en mm) : Total available water column stored in soil profile

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

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

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

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

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

**10 - Dr** -OUT- (en mm/d) : Deep drainage

procedure RempliRes(const Ru, ProfRu, RuSurf, EauDispo, StRurMax : Double; var stRu, stRUR, Hum, stRuSurf, Dr : Double) ;

{

Hypotheses :

On a une dynamique de l'eau represente par un remplissage par le haut

et une evolution des tailles de reservoirs quand ce remplissage est sup

a la quantite maximum de la taille en cours (front d'humectation).

Quand on a atteind la taille maximum par remplissage on considere

que c'est du drainage.

A l'interieur d'un reservoir l'eau est repartie de maniere homogene

(peu etre considere valable jusqu'a 2m de profondeur, dixit CB

d'apres d'autres sources).

3 representation des reservoirs permettant de simuler 3 reservoirs:

1)ensemble des reservoirs en eau, evoluant en profondeur en fonction

du front d'humectation

2) reservoir de surface (taille fixe)ou s'effectue l'evaporation et une part de la

transpiration en presence de racines

Modif : on a ajoute l'evaporation en dessous du seuil Pf4.2 estime a

la moitie de la RU.

3) reservoir contenant les racines, evoluant en fonction du front racinaire

REMARQUE : Ces reservoirs se chevauche

Au lieu de gerer des profondeurs on gere des stocks d'eau

(stRuMax stock d'eau maxi (RU \* prof. Max)}

var

StRuMax : Double;

StRuSurfPrec, EauTranspi : Double;

begin

try

StRuMax := Ru ;

StRuMax := StRuMax \* ProfRu / 1000;

// Rempli Res surface

StRuSurfPrec := StRuSurf;

StRuSurf:= min(StRuSurf+EauDispo , RuSurf + Rusurf / 10);

// enleve la qte d'eau remplissant la partie uniquement evaporable

if (StRuSurfPrec < Rusurf / 10) then

begin

EauTranspi := Max (0, EauDispo - (Rusurf / 10 - StRuSurfPrec));

end

else

begin

EauTranspi := EauDispo;

end;

// remplissage des autres réservoirs

// rempli res en eau, estime drainage, et evolution du front d'humectation

StRu := StRu + EauTranspi;

if (StRu > StRuMax) then

begin

Dr := StRu - StRuMax;

StRu := StRuMax;

end

else

begin

Dr := 0;

end;

Hum:= max(Hum, StRu);

// Rempli res racines

StRur:= min(StRur + EauTranspi, StRurMax);

StRur:= min(StRur, StRu);

except

AfficheMessageErreur('RempliRes',UBilEau);

end;

end;

**Module n°20 - BhyFESW** Procédure Delphi: EvalFESW, s'exécute tous les jours sur les entités concernées :

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

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

**3 - FESW** -OUT- (en m3/m3) : Fraction d'eau évaporable dans le sol

procedure EvalFESW(const stRuSurf, RuSurf : Double; var fesw : Double);

{

Estimation de la fraction d'eau evaporable, rapporte donc au reservoir

de surface, RuSurf est le stock d'eau maxi disponible pour la plante

sur ce reservoir

Modif : on considere que pour l'évaporation la moitie de cette

valeur doit etre ajoutée.

// Parametres

IN:

StRusurf : mm

RuSurf : mm

OUT:

fesw : mm

}

begin

try

fesw := StRuSurf / (RuSurf + Rusurf / 10);

except

AfficheMessageErreur('EvalFESW',UBilEau);

end;

end;

**Module n°21 - BhyKce** Procédure Delphi: EvalKce, s'exécute tous les jours sur les entités concernées :

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

**2 - Ltr** -IN- (en MJ/MJ) : Taux de rayonnement transmis au sol

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

procedure EvalKce(const Mulch, Ltr : Double; var Kce : Double);

begin

try

Kce := Mulch/100 \* ltr;

except

AfficheMessageErreur('EvalKce',UBilEau);

end;

end;

**Module n°22 - BhySoilWaterNeed** Procédure Delphi: DemandeSol, s'exécute tous les jours sur les entités concernées :

**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 DemandeSol(const Etp, Kce : Double; var evapPot : Double);

{

Estimation de l'evaporation potentielle du sol, on ne tient pas

compte d'une variation de l'evaporation en fonction d'une humectation

differente entre le haut et le bas du reservoir, on a un parametre

mulch qui peu traduire le phenomene d'auto mulching (defaut : 0.7)

qui peu aussi traduire un mulch par couverture vegetale ou...

La reduction de l'evaporation par l'evolution de la couverture

du sol par la plante est traduit par ltr.

// Parametres

IN:

ETo : mm

Kce : %

OUT:

evapPot : mm

}

begin

try

EvapPot:= ETP\* Kce;

except

AfficheMessageErreur('DemandeSol',UBilEau);

end;

end;

**Module n°23 - EvapRuSurfFesw** Procédure Delphi: EvapRuSurfFesw, s'exécute tous les jours sur les entités concernées :

L'evaporation est fonction de FESW seulement et pas au carré

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

**2 - FESW** -IN- (en m3/m3) : Fraction d'eau évaporable dans le sol

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

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

procedure EvapRuSurfFesw(const evapPot, fesw, stRuSurf : Double; var evap : Double);

{

Estimation de l'evaporation relle, rapporte a la fraction d'eau evaporable

// Parametres

IN:

fesw : mm

evapPot : mm

stRuSurf : mm

OUT:

evap : mm

}

begin

try

evap := EvapPot \* fesw;

evap := min(evap, StRuSurf);

except

AfficheMessageErreur('EvapRuSurfFesw',UBilEau);

end;

end;

**Module n°24 - BhyFTSW** Procédure Delphi: EvalFTSW, s'exécute tous les jours sur les entités concernées :

taux de remplissage de la RU Racinaire

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

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

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

procedure EvalFTSW(const RuRac,StockRac : Double; var ftsw : Double);

{

Estimation de la fraction d'eau transpirable, rapporte donc au reservoir

contenant les racines

// Parametres

IN:

RuRac : mm

StockRac : mm

OUT:

ftsw : mm

}

begin

try

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),UBilEau);

end;

end;

**Module n°25 - BhyKcp** Procédure Delphi: EvolKcp, s'exécute tous les jours sur les entités concernées :

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

**2 - Ltr** -IN- (en MJ/MJ) : Taux de rayonnement transmis au sol

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

procedure EvolKcp(const KcMax, Ltr : Double; var Kcp : Double);

begin

try

kcp := KcMax \* (1 - Ltr);

except

AfficheMessageErreur('EvolKcp',UBiomasse);

end;

end;

**Module n°26 - BhyCropWaterNeed** Procédure Delphi: DemandePlante, s'exécute tous les jours sur les entités concernées :

**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°27 - BhyKcTot** Procédure Delphi: EvalKcTot, s'exécute tous les jours sur les entités concernées :

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

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

**3 - Kc** -OUT- (en fraction) : Simulated crop coefficient Kc (Evaptranspiration / Eto) taking into account ground cover and water relations of soil and plant

procedure EvalKcTot(const Kce, Kcp : Double; var KcTot : Double);

begin

try

if (Kcp = Nullvalue) then

begin

Kctot := Kce;

end

else

begin

Kctot := Kce+ Kcp;

end;

except

AfficheMessageErreur('EvalKcTot',UBilEau);

end;

end;

**Module n°28 - BhyCstrPFactor** Procédure Delphi: CstrPFactor, s'exécute tous les jours sur les entités concernées :

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

**2 - 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

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

**4 - 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

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

procedure CstrPFactor(const Kcp, ParPFact, ftsw, Etp : Double; var cstr : Double);

var

pFact: Extended;

begin

try

pFact := ParPFact + 0.04 \* (5 - max(Kcp,1) \* ETP);

pFact := max(0.1, pFact);

pFact := min(0.8, pFact);

cstr := min((ftsw / (1 - pFact)), 1);

cstr := max(0, cstr);

except

AfficheMessageErreur('CstrPFactor | pFact: '+FloatToStr(pFact),UBilEau);

end;

end;

**Module n°29 - BhyTranspi** Procédure Delphi: EvalTranspi, s'exécute tous les jours sur les entités concernées :

**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°30 - BhyUpTakeResSep** Procédure Delphi: ConsoResSep, s'exécute tous les jours sur les entités concernées :

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

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

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

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

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

**6 - Tr** -INOUT- (en mm/d) : Actual crop transpiration

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

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

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

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

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

procedure ConsoResSep(const stRurMax, RuSurf, evap, trPot, evapPot : Double; var tr, stRur, stRuSurf, stRu, Etr, Etm : Double);

{

Separation de tr et evap. Consommation de l'eau sur les reservoirs

Hypothese : l'evaporation est le processus le plus rapide, retranche

en premier sur le reservoir de surface. Comme reservoir de surface

et reservoirs racinaires se chevauchent, il nous faut aussi calcule sur

le reservoir ayant des racines la part deja extraite pour l'evaporation.

Quand la profondeur des racines est inferieur au reservoir de surface

on ne consomme en evaporation que la fraction correspondant a cette

profondeur sur celle du reservoir de surface (consoRur).

Les estimations d'evaporation et de transpirations sont effectues

separemment, on peut ainsi avoir une consommation legerement superieure

a l'eau disponible. On diminuera donc la transpiration en consequence.

Modif : Pour les stock d'eau on tient compte de la partie rajoutee au

reservoir de surface qui ne peut etre que evapore (air dry)

// Parametres

IN:

stRurMax : mm

RuSurf : mm

evap : mm

trPot : mm

evaPot : mm

INOUT :

stRuSurf : mm

tr : mm

stRur : mm

stRu : mm

OUT:

etr : mm

etm : mm

}

var

ConsoRur, TrSurf : double;

begin

try

// part transpirable sur le reservoir de surface

TrSurf := max(0 ,StRuSurf - Rusurf / 10);

// qte d'eau evapore a consommer sur le reservoir de surface

StRuSurf := max(StRuSurf - Evap, 0);

// qte d'eau evapore a retirer sur la part transpirable

if (Evap > TrSurf) then

begin

ConsoRur:= TrSurf;

end

else

begin

ConsoRur:=Evap;

end;

StRu := Max(0, StRu - ConsoRur);

// fraction d'eau evapore sur la part transpirable qd les racines sont moins

// profondes que le reservoir de surface, mise a jour des stocks transpirables

if (stRurMax < RuSurf) then

begin

ConsoRur := Evap \* stRur / RuSurf;

end;

StRur := Max(0, StRur - ConsoRur);

// reajustement de la qte transpirable considerant que l'evap a eu lieu avant

// mise a jour des stocks transpirables

if (Tr > StRur) then

begin

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

end;

if (StRur > 0) then

begin

StRuSurf := max(StRuSurf - (Tr \* min(TrSurf / StRur, 1)), 0);

end;

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

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

Etr := tr + Evap;

Etm := trPot + Evap;

except

AfficheMessageErreur('ConsoResSep',UBilEau);

end;

end;

**Module n°31 - BilhyETRETM** Procédure Delphi: EvalETRETM, s'exécute tous les jours sur les entités concernées :

Evap + Transpi

**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°32 - EvalVitesseRacSarraV3** Procédure Delphi: EvalVitesseRacSarraV3, s'exécute tous les jours sur les entités concernées :

Vitesse rac par phase et test de valeur nulles

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

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

procedure EvalVitesseRacSarraV3(const VRacLevee, RootSpeedBVP, RootSpeedRPR, RootSpeedPSP, RootSpeedMatu1, RootSpeedMatu2, NumPhase : Double; var VitesseRacinaire : Double);

begin

try

case Trunc(NumPhase) of

1 : VitesseRacinaire := VRacLevee;

2 : VitesseRacinaire := RootSpeedBVP;

3 : VitesseRacinaire := RootSpeedPSP;

4 : VitesseRacinaire := RootSpeedRPR;

5 : begin

if (RootSpeedMatu1 = NullValue) then

begin

VitesseRacinaire := 0;

end

else

begin

VitesseRacinaire := RootSpeedMatu1;

end;

end;

6 : begin

if (RootSpeedMatu2 = NullValue) then

begin

VitesseRacinaire := 0;

end

else

begin

VitesseRacinaire := RootSpeedMatu2;

end;

end;

else

VitesseRacinaire := 0

end;

except

AfficheMessageErreur('EvalVitesseRacSarraV3 | NumPhase: '+FloatToStr(NumPhase),UPhenologie);

end;

end;

**Module n°33 - EvalLtr** Procédure Delphi: EvalLtr, s'exécute tous les jours sur les entités concernées :

**1 - 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

**2 - Lai** -IN- (en m²/m²) : leaf area index (green leaf blades only)

**3 - Ltr** -OUT- (en MJ/MJ) : Taux de rayonnement transmis au sol

procedure EvalLtr(const Kdf, Lai : Double; var ltr : Double);

begin

try

ltr := exp(-KDf \* lai) ;

except

AfficheMessageErreur('EvalLtr',UBiomasse);

end;

end;

**Module n°34 - EvalConversion** Procédure Delphi: EvalConversion, s'exécute tous les jours sur les entités concernées :

Calcule le coefficient b en fonction de Kassim de chaque phase.

**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 kg/ha/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°35 - EvalParIntercepte** Procédure Delphi: EvalParIntercepte, s'exécute tous les jours sur les entités concernées :

PAR intercepté journalier (fonction de LTR)

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

**2 - Ltr** -IN- (en MJ/MJ) : Taux de rayonnement transmis au sol

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

procedure EvalParIntercepte(const PAR, LTR : Double; var PARIntercepte : Double);

begin

try

PARIntercepte := PAR \* (1 - LTR);

except

AfficheMessageErreur('EvalParIntercepte | PAR: '+FloatToStr(PAR)+

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

end;

end;

**Module n°36 - EvalAssimPot** Procédure Delphi: EvalAssimPot, s'exécute tous les jours sur les entités concernées :

Assimilation potentielle

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

**2 - Conversion** -IN- (en kg/ha/MJ)

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

procedure EvalAssimPot(const PARIntercepte, Conversion : Double; var AssimPot : Double);

begin

try

AssimPot := PARIntercept e\* Conversion \* 10;

except

AfficheMessageErreur('EvalAssimPot | PAR Intercepté: '+FloatToStr(PARIntercepte)+

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

end;

end;

**Module n°37 - EvalAssimSarrahV3** Procédure Delphi: EvalAssimSarrahV3, s'exécute tous les jours sur les entités concernées :

Calcul des assimilats avec frein Tr/TrPot (concept flux de sève)

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

procedure EvalAssimSarrahV3(const AssimPot, Tr, TrPot : Double; var Assimilats : Double);

begin

try

if (TrPot > 0) then

begin

Assimilats := AssimPot \* Tr / TrPot;

end

else

begin

Assimilats := 0;

end;

except

AfficheMessageErreur('EvalAssimSarrahV3 | AssimPot: '+FloatToStr(AssimPot)+

' TrPot: '+FloatToStr(TrPot),UBilanCarbonSarra);

end;

end;

**Module n°38 - EvalRespMaint** Procédure Delphi: EvalRespMaint, s'exécute tous les jours sur les entités concernées :

**1 - BiomasseTotale** -IN- (en kg/ha) : Biomasse totale

**1 - BiomasseTotale** -IN- (en kg/ha) : Biomasse totale

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

**3 - TempMaint** -IN- (en °C)

**4 - TxRespMaint** -IN- (en kg/kg) : Taux de respiration de maintenance de la biomasse totale

**5 - DRespMaint** -OUT- (en kg/ha) : Quantité de sucre consommée pour la respiration de maintenance

**5 - DRespMaint** -OUT- (en kg/ha) : Quantité de sucre consommée pour la respiration de maintenance

procedure EvalRespMaint(const biomTot, tempMoy, KtempMaint, kRespMaint : Double; var respMaint : Double);

{

RespMaint Kg/ha/j en équivalent matière sèche

KRespMaint (0.015)

KTempMaint °C (25 )

}

begin

try

respMaint := kRespMaint \* biomTot \* power(2, (tempMoy - kTempMaint) / 10);

except

AfficheMessageErreur('EvalRespMaint',UBiomasse);

end;

end;

**Module n°39 - EvolBiomTotSarrahV3** Procédure Delphi: EvolBiomTotSarrahV3, s'exécute tous les jours sur les entités concernées :

Calcul biomasse Totale avec initialisation en début de phase 1

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

**2 - DRespMaint** -IN- (en kg/ha) : Quantité de sucre consommée pour la respiration de maintenance

**2 - DRespMaint** -IN- (en kg/ha) : Quantité de sucre consommée pour la respiration de maintenance

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

**4 - Densite** -IN- (en pieds/Ha) : Nombre de pieds /ha

**5 - TxResGrain** -IN- (en fraction) : Fraction of seed weight mibilizabme for growth of seeding

**6 - PoidsSecGrain** -IN- (en g) : Dry weight of single seed (or filled grain) in g, or 1000-grain dry wt in kg

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

**8 - BiomasseTotale** -INOUT- (en kg/ha) : Biomasse totale

**8 - BiomasseTotale** -INOUT- (en kg/ha) : Biomasse totale

**9 - DBiomTot** -OUT- (en kg/ha) : Biomasse totale créée dans la journée

**9 - DBiomTot** -OUT- (en kg/ha) : Biomasse totale créée dans la journée

procedure EvolBiomTotSarrahV3(const Assimilats, RespMaint, NumPhase, Densite, KResGrain, BiomasseGrain, ChangePhase : Double; var BiomasseTotale, DeltaBiomasseTotale : Double);

begin

try

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

begin

BiomasseTotale := Densite \* KResGrain \* BiomasseGrain / 1000;

end

else

begin

BiomasseTotale := BiomasseTotale + (Assimilats - RespMaint);

end;

DeltaBiomasseTotale := (Assimilats - RespMaint);

except

AfficheMessageErreur('EvolBiomTotSarrahV3 | BiomasseTotale: '+FloatToStr(BiomasseTotale)+

' DeltaBiomasseTotale: '+FloatToStr(DeltaBiomasseTotale),UBilanCarbonSarra);

end;

end;

**Module n°40 - EvalRdtPotRespSarrahV3** Procédure Delphi: EvalRdtPotRespSarrahV3, s'exécute tous les jours sur les entités concernées :

SI le RdtPot du jour est iau minimum égal au tiers de la respMaint

**1 - KRdtPotA** -IN- (en kg/kg) : Coeff d'évaluation du rendement potentiel (type HI)

**2 - BiomasseTotale** -IN- (en kg/ha) : Biomasse totale

**2 - BiomasseTotale** -IN- (en kg/ha) : Biomasse totale

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

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

**5 - KRdtPotB** -IN- (en kg/ha) : Constante d'évaluation du rendement potentiel

**5 - KRdtPotB** -IN- (en kg/ha) : Constante d'évaluation du rendement potentiel

**6 - KRdtBiom** -IN-

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

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

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

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

**11 - DRespMaint** -IN- (en kg/ha) : Quantité de sucre consommée pour la respiration de maintenance

**11 - DRespMaint** -IN- (en kg/ha) : Quantité de sucre consommée pour la respiration de maintenance

**12 - RdtPot** -INOUT- (en kg/ha) : Rendement potentiel de la culture évalué au stade de floraison

**12 - RdtPot** -INOUT- (en kg/ha) : Rendement potentiel de la culture évalué au stade de floraison

**13 - BiomasseTotaleStadeF** -OUT- (en kg/ha) : Biomasse totale au stade de floraison

**13 - BiomasseTotaleStadeF** -OUT- (en kg/ha) : Biomasse totale au stade de floraison

**14 - BiomTotStadeIP** -INOUT- (en kg/ha) : Biomasse totale au stade initiation paniculaire

**14 - BiomTotStadeIP** -INOUT- (en kg/ha) : Biomasse totale au stade initiation paniculaire

**15 - DRdtPot** -OUT- (en kg/ha) : Demande Potentielle journalière du rendement

**15 - DRdtPot** -OUT- (en kg/ha) : Demande Potentielle journalière du rendement

procedure EvalRdtPotRespSarrahV3(const KRdt, BiomasseTotale, NumPhase, ChangePhase, KrdtB, KRdtBiom, DegresDuJour, SeuilTempMatu1, tr, trPot, RespMaint : Double; var RdtPot, BiomTotStadeFloraison, BiomTotStadeIp, RdtPotDuJour : Double);

begin

try

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

begin

BiomTotStadeIp := BiomasseTotale;

end;

if (NumPhase = 5) then

begin

if (ChangePhase = 1) then

begin

BiomTotStadeFloraison := BiomasseTotale;

RdtPot := (KRdt \* (BiomTotStadeFloraison - BiomTotStadeIp) + KrdtB) + KRdtBiom \* BiomTotStadeFloraison;

end;

if (trpot > 0) Then

begin

RdtPotDuJour := RdtPot \* (DegresDuJour / SeuilTempMatu1) \* (tr / trPot);

RdtPotDuJour := max(RdtPotDuJour, RespMaint \* 0.15);

end

else

begin

RdtPotDuJour := 0;

end;

end ;

except

AfficheMessageErreur('EvalRdtPotRespSarrahV3',UBilanCarbonSarra);

end;

end;

**Module n°41 - EvolBiomAeroSarrahV3** Procédure Delphi: EvolBiomAeroSarrahV3, s'exécute tous les jours sur les entités concernées :

Evolution allométrique de la biomasse aérienne

**1 - BiomasseTotale** -IN- (en kg/ha) : Biomasse totale

**1 - BiomasseTotale** -IN- (en kg/ha) : Biomasse totale

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

**3 - AeroTotPente** -IN- (en kg/kg) : Pente de la relation de répartition biomasse aérienne/biomasse totale

**4 - AeroTotBase** -IN- (en kg/kg) : Base de la droite de répartition des biomasses aérienne-totale

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

**6 - DBiomTot** -IN- (en kg/ha) : Biomasse totale créée dans la journée

**6 - DBiomTot** -IN- (en kg/ha) : Biomasse totale créée dans la journée

**7 - DeltaBiomasseAerienn** -OUT- (en kg/ha) : Biomasse aérienne créée durant le jour courant

**7 - DeltaBiomasseAerienn** -OUT- (en kg/ha) : Biomasse aérienne créée durant le jour courant

**8 - BiomasseAerienne** -INOUT- (en kg/ha) : Biomasse aérienne

**8 - BiomasseAerienne** -INOUT- (en kg/ha) : Biomasse aérienne

procedure EvolBiomAeroSarrahV3(const BiomasseTotale, NumPhase, KPenteAero, KBaseAero, ChEvalSlaRapBiom2angePhase, DeltaBiomasseTotale : Double; var DeltaBiomasseAerienne, BiomasseAerienne : Double);

begin

try

DeltaBiomasseAerienne := BiomasseAerienne;

If ((NumPhase >= 2) and (NumPhase <= 4)) then

begin

BiomasseAerienne := min(0.9, kPenteAero \* BiomasseTotale + kBaseAero) \* BiomasseTotale ;

end

else

begin

BiomasseAerienne := BiomasseAerienne + DeltaBiomasseTotale;

end;

DeltaBiomasseAerienne := BiomasseAerienne - DeltaBiomasseAerienne;

except

AfficheMessageErreur('EvolBiomAeroSarrahV3 | DeltaBiomasseAerienne: '+FloatToStr(DeltaBiomasseAerienne)+

' BiomasseAerienne: '+FloatToStr(BiomasseAerienne),UBilanCarbonSarra);

end;

end;

**Module n°42 - EvalReallocationSarrahV3** Procédure Delphi: EvalReallocationSarrahV3, s'exécute tous les jours sur les entités concernées :

Evalaution de la réallocation et du rendement potentiel pour la journée avec frein hydrique (tr/trpot)

**1 - DRdtPot** -IN- (en kg/ha) : Demande Potentielle journalière du rendement

**1 - DRdtPot** -IN- (en kg/ha) : Demande Potentielle journalière du rendement

**2 - DeltaBiomasseAerienn** -IN- (en kg/ha) : Biomasse aérienne créée durant le jour courant

**2 - DeltaBiomasseAerienn** -IN- (en kg/ha) : Biomasse aérienne créée durant le jour courant

**3 - TxRealloc** -IN- (en kg/kg) : Pourcentage de réallocation de carbone par rapport au manque en assimilat (Demande-Offre)

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

**5 - BiomasseFeuilles** -IN- (en kg/ha) : Biomasse des feuilles

**5 - BiomasseFeuilles** -IN- (en kg/ha) : Biomasse des feuilles

**6 - Reallocation** -OUT- (en kg/ha) : Quantité de réserves réallouées au rendement

**6 - Reallocation** -OUT- (en kg/ha) : Quantité de réserves réallouées au rendement

**7 - ManqueAssim** -OUT- (en kg/ha) : Déficit en assimilats (demande - offre)

**7 - ManqueAssim** -OUT- (en kg/ha) : Déficit en assimilats (demande - offre)

procedure EvalReallocationSarrahV3(const RdtPotDuJour, DeltaBiomasseAerienne, TxRealloc, NumPhase, Biomleaf : Double; var Reallocation, ManqueAssim : Double);

{

La reallocation est à 0 quand la biomasse foliaire verte est inférieure à 30 kh/ha

}

begin

try

if (NumPhase = 5) then

begin

ManqueAssim := max(0, (RdtPotDuJour - max(0, DeltaBiomasseAerienne)));

Reallocation := min(ManqueAssim \* TxRealloc, max (0, biomleaf - 30));

end

else

Begin

ManqueAssim := 0;

Reallocation := 0;

end;

except

AfficheMessageErreur('EvalReallocationSarrahV3 | RdtPotDujour: '+FloatToStr(RdtPotDujour)+

' DeltaBiomasseAerienne: '+FloatToStr(DeltaBiomasseAerienne)+

' TxRealloc: '+FloatToStr(TxRealloc)+

' NumPhase: '+FloatToStr(NumPhase),UBilanCarbonSarra);

end;

end;

**Module n°43 - EvalBiomasseRacinair** Procédure Delphi: EvalBiomasseRacinair, s'exécute tous les jours sur les entités concernées :

Biomasse racinaire

**1 - BiomasseTotale** -IN- (en kg/ha) : Biomasse totale

**1 - BiomasseTotale** -IN- (en kg/ha) : Biomasse totale

**2 - BiomasseAerienne** -IN- (en kg/ha) : Biomasse aérienne

**2 - BiomasseAerienne** -IN- (en kg/ha) : Biomasse aérienne

**3 - BiomasseRacinaire** -OUT- (en kg/ha) : Biomasse des racines

**3 - BiomasseRacinaire** -OUT- (en kg/ha) : Biomasse des racines

procedure EvalBiomasseRacinair(const BiomasseTotale, BiomasseAerienne : Double; var BiomasseRacinaire : Double);

begin

try

BiomasseRacinaire := BiomasseTotale - BiomasseAerienne;

except

AfficheMessageErreur('EvolBiomasseRacinair | BiomasseTotale: '+FloatToStr(BiomasseTotale)+

' BiomasseAerienne: '+FloatToStr(BiomasseAerienne)+

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

end;

end;

**Module n°44 - EvalFeuilleTigeSarrahV3** Procédure Delphi: EvalFeuilleTigeSarrahV3, s'exécute tous les jours sur les entités concernées :

Calcul des biomasses feuilles, relation allométrique avec diminution des biomasses feuilles quand diminution de biommasse aérienne. Biomasse Tige est ce qui reste de la biomasse aérienne

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

**2 - FeuilAeroBase** -IN- (en %) : Base de la droite de répartition des biomasses feuilles-aérienne

**3 - FeuilAeroPente** -IN- : Pente de la relation de répartition biomasse feuilles/biomasse aérienne

**4 - BiomasseAerienne** -IN- (en kg/ha) : Biomasse aérienne

**4 - BiomasseAerienne** -IN- (en kg/ha) : Biomasse aérienne

**5 - PcReallocFeuille** -IN-

**6 - Rdt** -IN- (en kg/ha) : Rendement en grains

**6 - Rdt** -IN- (en kg/ha) : Rendement en grains

**7 - DeltaBiomasseAerienn** -IN- (en kg/ha) : Biomasse aérienne créée durant le jour courant

**7 - DeltaBiomasseAerienn** -IN- (en kg/ha) : Biomasse aérienne créée durant le jour courant

**8 - Reallocation** -IN- (en kg/ha) : Quantité de réserves réallouées au rendement

**8 - Reallocation** -IN- (en kg/ha) : Quantité de réserves réallouées au rendement

**9 - BiomasseFeuilles** -INOUT- (en kg/ha) : Biomasse des feuilles

**9 - BiomasseFeuilles** -INOUT- (en kg/ha) : Biomasse des feuilles

**10 - BiomasseTiges** -OUT- (en kg/ha) : Biomasse de la tige

**10 - BiomasseTiges** -OUT- (en kg/ha) : Biomasse de la tige

**11 - DayBiomLeaf** -OUT- (en kg/ha) : Augmentation journalière de la biomasse foliaire

**11 - DayBiomLeaf** -OUT- (en kg/ha) : Augmentation journalière de la biomasse foliaire

procedure EvalFeuilleTigeSarrahV3(const NumPhase, kBaseLaiDev, kPenteLaiDev, BiomasseAerienne, PartFeuillesTiges, rdt, DeltaBiomasseAerienne, Reallocation : Double; var BiomasseFeuilles, BiomasseTiges, DeltaBiomasseFeuilles : Double);

var

bM, CM : Double;

begin

try

if (NumPhase > 1) then

begin

DeltaBiomasseFeuilles := BiomasseFeuilles;

if (DeltaBiomasseAerienne < 0) then

begin

BiomasseFeuilles := max(0, BiomasseFeuilles - (-DeltaBiomasseAerienne + Reallocation) \* PartFeuillesTiges);

end

else

begin

if (NumPhase <= 4) then

begin

bM := kBaseLaiDev - 0.1;

cM := ((kPenteLaiDev \* 1000)/ bM + 0.78) / 0.75;

BiomasseFeuilles := (0.1 + bM \* power(cM, BiomasseAerienne / 1000)) \* BiomasseAerienne;

end

else

begin

BiomasseFeuilles := BiomasseFeuilles - Reallocation \* PartFeuillesTiges;

end;

end;

BiomasseTiges:= BiomasseAerienne - (BiomasseFeuilles + rdt) ;

DeltaBiomasseFeuilles := BiomasseFeuilles - DeltaBiomasseFeuilles;

end;

except

AfficheMessageErreur('EvalFeuilleTigeSarrahV3',UBilanCarbonSarra);

end;

end;

**Module n°45 - EvalBiomasseVegetati** Procédure Delphi: EvalBiomasseVegetati, s'exécute tous les jours sur les entités concernées :

Feuilles + Tiges

**1 - BiomasseTiges** -IN- (en kg/ha) : Biomasse de la tige

**1 - BiomasseTiges** -IN- (en kg/ha) : Biomasse de la tige

**2 - BiomasseFeuilles** -IN- (en kg/ha) : Biomasse des feuilles

**2 - BiomasseFeuilles** -IN- (en kg/ha) : Biomasse des feuilles

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

**4 - BiomasseVegetative** -OUT- (en kg/ha) : Biomasse végétative (racine, feuilles, tige)

**4 - BiomasseVegetative** -OUT- (en kg/ha) : Biomasse végétative (racine, feuilles, tige)

procedure EvalBiomasseVegetati(const BiomasseTiges, BiomasseFeuilles, NumPhase : Double; var BiomasseVegetative : Double);

begin

try

BiomasseVegetative := BiomasseTiges +BiomasseFeuilles;

except

AfficheMessageErreur('EvolBiomasseVegetati | BiomasseTiges: '+FloatToStr(BiomasseTiges)+

' BiomasseFeuille: '+FloatToStr(BiomasseFeuilles),UMilBilanCarbone);

end;

end;

**Module n°46 - EvalSlaSarrahV3** Procédure Delphi: EvalSlaSarrahV3, s'exécute tous les jours sur les entités concernées :

Evolution dynamique du SLA en fn de la dynamique de la bimasse foliaire

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

**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

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

**3 - SlaPente** -IN- : Dépend du module utilisé

**4 - DayBiomLeaf** -IN- (en kg/ha) : Augmentation journalière de la biomasse foliaire

**4 - DayBiomLeaf** -IN- (en kg/ha) : Augmentation journalière de la biomasse foliaire

**5 - BiomasseFeuilles** -IN- (en kg/ha) : Biomasse des feuilles

**5 - BiomasseFeuilles** -IN- (en kg/ha) : Biomasse des feuilles

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

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

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

procedure EvalSlaSarrahV3(const SlaBVP, SlaRPR, KpenteSla, DeltaBiomasseFeuilles, BiomasseFeuilles, NumPhase, ChangePhase : Double; var sla : Double);

{

On suppose que les jeunes feuilles on un SLA supérieur aux vieilles feuilles.

La fraction de jeunes (nouvelles) feuilles fait donc monter le SLA global

du couvert. Le paramètre penteSLA provoque une chute générale du SLA

(penteSLA = chute relative par jour = fraction de différence entre SLAmax

et SLAmin). Fonctionnement conçu surtout pour les légumineuses, mais

peut être aussi adapté aux autres espèces.

Paramètres :

SLAmax (0.001 … 0.01), ex : 0.007

SLAmin (0.001 … 0.01), ex : 0.002

penteSLA (0 … 0.2), ex : 0.1

Avec : SLAini = SLAmax

}

begin

try

if (BiomasseFeuilles > 0) then

begin

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

begin

sla := SlaBvp;

end;

sla := (sla - KpenteSla \* (sla - SlaRPR)) \* (BiomasseFeuilles - DeltaBiomasseFeuilles) / BiomasseFeuilles + (SlaBVP + sla)/2 \* (DeltaBiomasseFeuilles / BiomasseFeuilles);

sla := min(SlaBVP, max(SlaRPR, sla));

end;

except

AfficheMessageErreur('EvalSlaSarrahV3',UBilanCarbonSarra);

end;

end;

**Module n°47 - EvolLAIPhases** Procédure Delphi: EvolLAIPhases, s'exécute tous les jours sur les entités concernées :

Calcul LAI = sla x BiomFeuilles avec LAI = 0 avant levée

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

**2 - BiomasseFeuilles** -IN- (en kg/ha) : Biomasse des feuilles

**2 - BiomasseFeuilles** -IN- (en kg/ha) : Biomasse des feuilles

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

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

procedure EvolLAIPhases(const NumPhase, BiomasseFeuilles, sla : Double; var LAI : Double);

begin

try

if (NumPhase <= 1) then

begin

LAI := 0

end

else

begin

if (NumPhase <= 6) then

begin

LAI:= BiomasseFeuilles \* sla;

end

else

begin

LAI:=0;

end;

end;

except

AfficheMessageErreur('EvolLAIPhases: ',UMilBilanCarbone);

end;

end;

**Module n°48 - EvolDayRdtSarraV3** Procédure Delphi: EvolDayRdtSarraV3, s'exécute tous les jours sur les entités concernées :

Evolution journalière Fn degré jour et RdtPot avec la contrainte Tr/Trpot, notion de frein lié a réduction du flux de sève

**1 - DRdtPot** -IN- (en kg/ha) : Demande Potentielle journalière du rendement

**1 - DRdtPot** -IN- (en kg/ha) : Demande Potentielle journalière du rendement

**2 - DeltaBiomasseAerienn** -IN- (en kg/ha) : Biomasse aérienne créée durant le jour courant

**2 - DeltaBiomasseAerienn** -IN- (en kg/ha) : Biomasse aérienne créée durant le jour courant

**3 - Reallocation** -IN- (en kg/ha) : Quantité de réserves réallouées au rendement

**3 - Reallocation** -IN- (en kg/ha) : Quantité de réserves réallouées au rendement

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

**5 - Rdt** -INOUT- (en kg/ha) : Rendement en grains

**5 - Rdt** -INOUT- (en kg/ha) : Rendement en grains

procedure EvolDayRdtSarraV3(const RdtPotDuJour, DeltaBiomasseAerienne, Reallocation, NumPhase : Double; var Rdt : Double);

{

On tend vers le potentiel en fn du rapport des degresJours/sumDegresJours

pour la phase de remplissage

Frein sup fn du flux de sève estimé par le rapport Tr/TrPot

}

begin

try

if (NumPhase = 5) then

begin

Rdt := Rdt + min(RdtPotDuJour, Max(0, DeltaBiomasseAerienne) + Reallocation);

end;

except

AfficheMessageErreur('EvolDayRdtSarraV3',UBilanCarbonSarra);

end;

end;

**Module n°49 - PhotoperSarrahV3** Procédure Delphi: PhotoperSarrahV3, s'exécute tous les jours sur les entités concernées :

Photoperiode concept impatience, intégration test de fin de photopériode

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

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

**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

**9 - 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

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

**12 - PhasePhotoper** -INOUT- (en Numerique) : Egal a 1 pendant la phase photper, 0 sinon

procedure PhotoperSarrahV3(const Numphase, ChangePhase, SomDegresJour, DegresDuJour, SeuilPP, PPCrit, DureeDuJour, PPExp, PPSens : Double; var SumPP, SeuilTempPhasePrec, PhasePhotoper : Double);

{Procedure speciale Vaksman Dingkuhn valable pour tous types de sensibilite

photoperiodique et pour les varietes non photoperiodique.

PPsens varie de 0,4 a 1,2. Pour PPsens > 2,5 = variété non photoperiodique.

SeuilPP = 13.5

PPcrit = 12

SumPP est dans ce cas une variable quotidienne (et non un cumul)}

begin

try

if (NumPhase = 3) then

begin

if (ChangePhase = 1) then

begin

SumPP := 100;

end;

SumPP := power((1000 / max(0.01, SomDegresJour - SeuilTempPhasePrec)), PPExp) \* max(0, (DureeDuJour - PPCrit)) / (SeuilPP - PPCrit);

if (sumPP <= PPsens) then

begin

PhasePhotoper :=0;

end;

end;

except

AfficheMessageErreur('PhotoperSarrahV3',UPhenologie);

end;

end;

**Module n°50 - MortaliteSarraV3** Procédure Delphi: MortaliteSarraV3, s'exécute tous les jours sur les entités concernées :

Test de mortalité de la plante après semis

**1 - DeltaBiomasseAerienn** -IN- (en kg/ha) : Biomasse aérienne créée durant le jour courant

**1 - DeltaBiomasseAerienn** -IN- (en kg/ha) : Biomasse aérienne créée durant le jour courant

**3 - SeuilCstrMortality** -IN- (en d) : Sets the cumulative, uninterrupted drought necessary to kill the plant (simulation ends)

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

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

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

procedure MortaliteSarraV3(const DeltaBiomasseAerienne, NbjTestSemis,SeuilCstrMortality, ChangePhase : Double; var NumPhase, RuRac : Double);

{

Test sur 20 jours

Dès que le delta est négatif sur 10 jours

}

begin

try

if (NumPhase >= 2) then

begin

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

begin

NbjourCompte := 0;

NbjStress := 0;

end;

inc(Nbjourcompte);

if (Nbjourcompte < NbjTestSemis) then

begin

if (DeltaBiomasseAerienne < 0) then

begin

inc(NbjStress);

end;

end;

if (NbjStress > SeuilCstrMortality) then

begin

Numphase := 0;

RuRac := 0;

NbJStress := 0;

end;

end;

except

AfficheMessageErreur('MortaliteSarraV3',UBilanCarbonSarra);

end;

end;