PLANT RESEARCH INTERNATIONAL
WAGENINGEN UR

Beschrijving van TIPSTAR

Hét simulatiemodel voor groei en productie van zetmeelaardappelen

Vertrouwelijk

Don M. Jansen (Ed)

Nota 547

PLANT RESEARCH INTERNATIONAL
WAGENINGEN UR

Beschrijving van TIPSTAR

Hét simulatiemodel voor groei en productie van zetmeelaardappelen

Vertrouwelijk

Don M. Jansen (Ed)

Plant Research International B.V., Wageningen
Juni 2008 Nota 547

Plant Research International B.V.

Adres : Droevendaalsesteeg 1, Wageningen
: Postbus 16, 6700 AA Wageningen
Tel. : 0317 – 48 60 01
Fax : 0317 – 41 80 94
E-mail : info.pri@wur.nl
Internet : www.pri.wur.nl

Inhoudsopgave

pagina

1. Inleiding

Don Jansen

Tipstar in ontwikkeld voor toepassing in onderzoek en kennisoverdracht ten behoeve van de teelt van zetmeel-aardappelen. Het bevat

1. een gewassimulatiemodule waarmee groei en ontwikkeling van een gewas zetmeelaardappelen te berekenen is. Deze module bouwt voort op ideeën van Daniel van Kraalingen over hoe knolgroei te simuleren (Kraalingen, pers. comm). Ten opzichte van zijn voorganger (Haren & Jansen, 1999) is het uitgebreid met de mogelijkheid om effecten van stikstofgebrek te berekenen. Het gebruikt en is gevoelig voor de waarde van situatiespecifieke gegevens (Jansen et al., 2003b) die kenmerken beschrijven van
   a. de gebruikte cultivar, zoals snelheid van bladvorming, parameters om de relatie tussen de hoeveelheid droge stof in de knollen en de gevormde hoeveelheid zetmeel
   b. het weer waaronder de cultivar geteeld wordt, zoals het verloop in de tijd van maximale en minimale temperatuur, straling, regen
   c. de bodem waarop de cultivar geteeld wordt, zoals diepte van het grondwater, parameters om de bodemfysische en -chemische karakteristieken van de bodem te beschrijven
   d. het management, zoals datum van opkomst, datum, type en hoeveelheid van toediening van stikstofbemesting
2. modules waarmee het effect van bepaalde veranderingen in teelt en bewaring berekend kan worden; een gebruiker kan hiermee analyseren welke verbeteringen in teelt en bewaring interessant zouden zijn. Deze modules maken het beter mogelijk om TIPSTAR in te zetten voor kennisoverdracht.
3. een procedure en bijbehorende software modules om het simulatiemodel te calibreren op één of meerdere sets van waarnemingen; een voorbeeld van een dergelijke calibratie wordt gegeven door Jansen, 2002b.
4. een procedure en bijbehorende software modules om met het simulatiemodel een optimale vorm (tijdstip en hoeveelheid) van stikstoftoediening en/of beregening te berekenen voor een specifieke situatie (ras, tijdstip van opkomst, bodemtype, weerscondities) binnen door de gebruiker op te geven randvoorwaarden. De optimalisatie kan uitgevoerd worden voor operationele beslissingen, bijv. of er vandaag of volgende week beregend zou moeten worden, en voor tactische beslissingen, bijv. wat gemiddeld genomen de beste bemestingsstrategie zou zijn. Een voorbeeld van toepassing in operationele beslissingen wordt gegeven door Jansen et al., 2003a en Klok & Wustman, 2002.

Technische details van Tipstar en een handleiding voor het gebruik ervan staan beschreven in Jansen (2002a). In het onderhavige rapport worden de inhoudelijke aspecten van de gewassimulatie module beschreven.
In hoofdstuk 2 wordt eerst een overzicht gegeven van verschillende sub-modules binnen TIPSTAR waaruit het simulatiemodel gevormd wordt en die het mogelijk maken om deze modules aan te sturen en onderling te laten communiceren. De inhoud van diverse simulatie modules wordt daarna in aparte hoofdstukken behandeld.

2. Opzet TIPSTAR

Don Jansen & Jacques Davies

2.1 Inleiding

TIPSTAR is opgebouwd uit modules die elk een specifieke taak uitvoeren. Die specifieke taak kan bestaan uit centrale sturing (Combctl), het inlezen en doorsturen van data (zoals CropDataModule) of het simuleren van onderdelen van de gewasgroei en -ontwikkeling waarin deze data gebruikt worden (zoals CropPosModule). Bij een aantal modules behoren datafiles die specifieke informatie bevatten betreffende de taak van die module. De gewassimulatiemodule bestaat uit een aantal sub-modules waarmee gewasgroei, effecten van bewaring op knolgewicht en -kwaliteit en een inschatting van kosten van bepaalde veranderingen ten opzichte van standaard gewas management gemaakt kunnen worden.

De gebruiker van TIPSTAR kan aangeven welke modules en /of datafiles in de simulatie te gebruiken of juist uit te sluiten via data-files die gebruikt worden in de centrale sturingsmodule. Bij simulatie, calibratie en optimalisatie is er de keuze om te rekenen met de potentiële groei van het gewas, waarin géén water of stikstofgebrek een rol speelt, of met gewasgroei die afhankelijk is van beschikbare hoeveelheid water (watergelimiteerd) of van hoeveelheid water en stikstof (stikstofgelimiteerd). Bij simulatie van zowel water- als stikstofgelimiteerde gewasgroei wordt voor simulatie van de dynamiek van water in de bodem een waterbalans module aangeschakeld, en van stikstofgelimiteerde gewasgroei ook een module voor de simulatie van de C en N balans in de bodem. Het is niet mogelijk om alleen effecten van stikstof.beschikbaarheid mee te nemen aangezien de bodemprocessen met betrekking tot stikstof sterk afhankelijk zijn van de processen die het bodemvocht betreffen.

Een belangrijk principe in TIPSTAR is dat uitwisseling van gegevens tussen modules plaats vindt door het plaatsen en lezen van parameters en variabelen (naam én waarde) op een 'Schoolbord' (ook 'blackboard' genoemd). Hiermee is het vrij makkelijk om verschillende submodellen voor één specifieke taak binnen TIPSTAR te kunnen aanbieden zonder andere modules te hoeven aanpassen. Zo is het binnen TIPSTAR mogelijk om verschillende bodem-water-modules te gebruiken (zoals SAWAH en SAHEL) door in die modellen de hoeveelheid water per laag op een gestandaardiseerde manier op het schoolbord te zetten, waarna deze gegevens door het water-opname submodel ingelezen worden.
Het inlezen van de situatiebeschrijvende parameters (muv weersgegevens) gebeurt via specifieke inleesmodules die de gegevens op het schoolbord zetten. Bij initialisatie van de simulatiemodules lezen deze de gegevens van het schoolbord af. Dit maakt het mogelijk om calibratie en optimalisatie te doen doordat tussen het wegschrijven van de parameters op het schoolbord enerzijds en het aflezen ervan door de simulatiemodules anderzijds een 'parameter-veranderings' module gezet kan worden (SetSpecParamsModule).

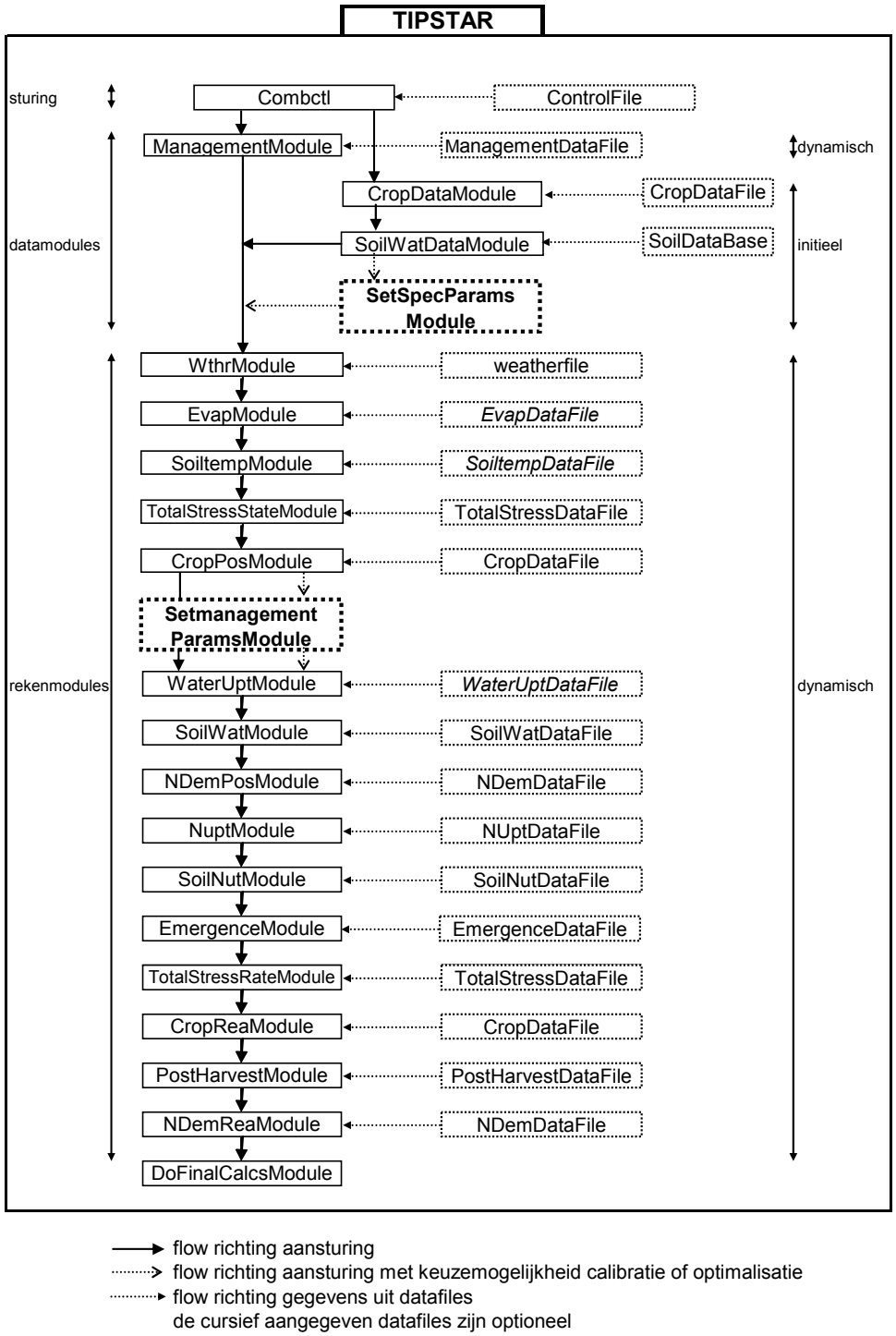
Een belangrijk principe in TIPSTAR is dat voor de berekening van de gewasgroei éérst de 'mogelijke' (Engels: 'possible') dagelijkse snelheden van groei en verdeling over de organen wordt berekend en daarna pas de 'gerealiseerde' (Engels: realized) snelheden. De 'mogelijke' snelheden zijn afhankelijk van de status van het gewas aan het begin van de dag en de weersomstandigheden op die dag. Het effect van de status van het gewas op de verschillende groei en ontwikkelingsprocessen wordt in een specifieke 'stress' module (TotalStress**State**Module) berekend voorafgaande aan de snelheidsberekening. Zo wordt bijvoorbeeld het effect van het stikstofgehalte in de bladeren op de fotosynthese in de TotalStressStateModule bepaald en als reductiefactor doorgegeven aan de Crop**Pos**Module waarin de mogelijke snelheden worden berekend. Deze mogelijke snelheden zijn dus in principe wat anders dan de 'potentiële' snelheden, die namelijk aangeven wat potentieel mogelijk zou zijn in afwezigheid van stressfactoren. Alleen wanneer de potentiële groei wordt gesimuleerd. i.e. wanneer effecten water-tekorten en/of -overschotten en N gebrek zijn uitgesloten, zullen de 'mogelijke' snelheden overeenkomen met de 'potentiële' groeisnelheden.
De 'mogelijke' snelheden worden doorgegeven aan submodellen waarmee de gerealiseerde wateropname (in WaterUptModule) en de met de mogelijke groei gepaard gaande vraag aan stikstof (NdemPosModule) wordt

berekend. De vraag aan stikstof wordt doorgegeven aan een submodel waarmee de gerealiseerde opname van stikstof wordt berekend (NuptModule) in afhankelijkheid van mogelijke vraag en de actuele situatie met betrekking tot stikstof in de bodem. Deze gerealiseerde opnames van water en stikstof kunnen minder zijn dan nodig is om de 'mogelijke' groei en ontwikkeling mogelijk te maken. Daartoe worden in de TotalStresss**Rate**Module de effecten van het verschil tussen 'mogelijke' en gerealiseerde snelheden van opname van water en stikstof op de groei en ontwikkelingsprocessen berekend. Deze effecten worden afgewogen (bijv. optelling van afzonderlijke effecten of bepaling van maximale effect) en als reductiefactoren op de verschillende 'mogelijke' groei en ontwikkelingssnelheden doorgestuurd naar de Crop**Rea**Module. Hierin wordt de gerealiseerde groei en ontwikkeling berekend en geïntegreerd over tijd zodat een gerealiseerde status van het gewas wordt verkregen. Deze gerealiseerde status van het gewas wordt in de berekeningen voor de volgende dag weer gebruikt in de TotalStressStateModule.
Naast effecten van stikstof en water, kunnen ook andere groei- en ontwikkelingbeperkende of belemmerende factoren meegenomen worden, zoals Phytophthora (cf Haren & Jansen, 1999). Ook dan worden in de TotalStressStateModule en TotalStressRateModule de effecten van aantasting door deze factoren berekend en afgewogen/geïntegreerd. Deze manier van werken heeft als voordeel, dat in de gewasgroei submodellen het niet nodig is om voor iedere afzonderlijke factor die meegenomen zou kunnen worden specifieke reductiefactoren op te nemen. Hierdoor is het mogelijk om aan- en afkoppeling van door te berekenen stress factoren simpel te regelen. Ook is het mogelijk om bij dezelfde aannames over effecten van die factoren verschillende gewassen door te rekenen, of, andersom, met één set van mogelijke en gerealiseerde gewasgroei submodellen meerdere gewassen door te rekenen waarbij alleen de stresseffecten eventueel anders berekend dienen te worden.

Om bovenstaande mogelijk te maken, moeten de diverse modules in TIPSTAR in een vaste volgorde aangeroepen worden. Dit is geregeld in de stuur-module Combctl. TIPSTAR werkt binnen de Fortran Simulation Environment schil (FSE, versie 3.3; Jongschaap, 1996; Kraalingen, 1995) en maakt gebruik van de TTUTIL software bibliotheek (Kraalingen & Rappoldt, 2000).

2.2 Korte beschrijving van de modules

In deze paragraaf wordt de volgorde van aanroepen en de functies van de verschillende modules beschreven. In Figuur 1 is de samenhang tussen de modules schematisch weergegeven.



Figuur 1. Volgorde van aanroepen en relaties tussen verschillende modules in TIPSTAR.

2.2.1 Combctl

Deze module heeft een centrale aansturende functie in TIPSTAR. Vanuit dit onderdeel worden de in Combctl.dat benoemde en geselecteerde modules en datafiles in een vastgelegde volgorde aangeroepen om het groeiproces van het gewas te simuleren, om parameters te calibreren of om management te optimaliseren. Combctl leest hiertoe de door de gebruiker toegewezen specifieke (i.e. namen van bestaande) simulatiemodules en datafiles, o.a. aan onderstaande generieke modules. Voor details zie Jansen2002a.

2.2.2 ManagementModule

De werkzaamheden die uitgevoerd worden m.b.t. de teelt van het gewas staan voor elke specifieke door te rekenen situatie in een specifieke datafile waarvan de naam via de generieke naam managementDatafile door de gebruiker gedefinieerd wordt. In deze file staan met name tijdstippen van poten, bemesten, oogsten en beregenen alsmede om het type en de toegepaste hoeveelheden meststoffen. De ManagementModule heeft tot taak gegevens van deze werkzaamheden op het juiste moment beschikbaar te maken door ze op het 'Schoolbord' te zetten zodat modules die van de gegevens gebruik maken deze ook kunnen verkrijgen. Zo wordt op de dag van bemesting de hoeveelheid toegediende anorganische en organische N op het Schoolbord gezet en zal de module SoilNutModule ze van het Schoolbord aflezen. In situaties waarin bemest wordt dan wel biociden gebruikt worden heeft de ManagementModule ook nog gegevens nodig betreffende de meststoffen (zoals N gehalte, % droge stof) en de biociden (% werkzame stof, duur werkzaamheid e.d.). Deze informatie wordt in de door de gebruiker te benoemen FertilizerDataFile en PesticideDataFile opgeslagen en in de ManagementModule ingelezen en op het 'Schoolbord' gezet.

2.2.3 CropDataModule

De CropDataModule is een module die de specifieke gewasgegevens, zoals drogestof -verdeling en groeisnelheden, in een database voor aardappelraseigenschappen opzoekt en voor gebruik in het model inleest. De gebruiker moet de juiste file-naam via CropDataFile doorgeven.

2.2.4 SoilWatDataModule

De module zoekt in een database de fysische bodemeigenschappen van het bodemprofiel op. Aan de hand van vooraf ingestelde keuzes kunnen per bodemlaag karakteristieke eigenschappen van het bodemprofiel berekend worden. De gebruiker geeft via SoilWatDataBase de naam van deze database file op. Voor specifieke settings van het gebruikte bodemwater submodel (gekozen via SoilWatModule) moet de gebruiker een datafile toewijzen aan SoilWatDataFile.

2.2.5 SetSpecParamsModule

Wanneer het model gebruikt moet worden in de calibratie- of optimalisatiemodus moet deze module geactiveerd worden. Zie Jansen 2002a voor details.

2.2.6 WthrModule

De weersgegevens worden door dit gedeelte ingelezen uit weerdatafiles en zonodig omgerekend en ingevoerd in het model.

2.2.7 EvapModule

In deze module wordt aan de hand van door Makkink opgestelde rekenregels de waterverdamping van het gewas en van de bodem berekend.

2.2.8 SoiltempModule

De SoiltempModule genereert voor elk in het bodemprofiel geselecteerde bodemlaag een temperatuur die afgeleid wordt van de gemiddelde dagtemperatuur van de lucht.

2.2.9 TotalStressStateModule

Deze module heeft de functie om een groeireductiefactor te berekenen voor elk plantorgaan dat een tekort aan water, stikstof of fosfor heeft. De groeireductiefactor kan ook afhankelijk zijn van een opgelopen plantenziekte.

2.2.10 CropPosModule

Deze module bevat rekenregels om de potentiële gewasgroei vanaf de opkomst te genereren. Er wordt geen rekening gehouden met water en/of stikstof tekort.

2.2.11 SetManagementParamsModule

In de keuze om bijv. een managementscenario te optimaliseren moet deze module geactiveerd worden. Tevens moet een set van voorwaarden ingevoerd worden waarbij het beste managementscenario moet worden berekend.

2.2.12 WaterUptModule

Deze module berekent de benodigde wateropname van het gewas met gewasparameters zoals verdamping, groeisnelheid en bewortelingsdiepte. Een mogelijk watertekort wordt doorgerekend naar het groeiproces van het gewas.

2.2.13 SoilWatModule

Deze module berekent de waterhuishouding in het bodemprofiel. Er worden berekeningen verricht aan de toe- en afname van de hoeveelheid water in de bodem door o.a. neerslag, opname gewas, verdamping en uitspoeling naar diepere bodemlagen. Voor Tipstar zijn twee verschillende modules beschikbaar, Sahel en Sawah, die elk op een karakteristieke wijze de waterhuishouding tussen bodem en plant benaderen.

2.2.14 NdemPosModule

In NdemPosModule wordt de minimale stikstofopname berekend die het gewas bij minimale potentiële groei nodig heeft.

2.2.15 NuptModule

Deze module berekent de benodigde stikstofopname door de plant per bodemlaag. Wanneer een tekort aan stikstof wordt geconstateerd dan heeft dat zijn weerslag op de productie van het gewas.

2.2.16 SoilNutModule

Deze module simuleert de omzettingsprocessen van organische stof en (delen van) meststoffen naar door de plant op te nemen stikstof in het bodemprofiel. Afhankelijk van de CN verhouding van de meststoffen vindt mineralisatie of immobilisatie van stikstof plaats.

2.2.17 EmergenceModule

Indien geactiveerd, wordt in deze module de verwachte opkomst van de gepote aardappelen berekend. De opkomst wordt met twee kenmerken beschreven: het aantal opgekomen aardappelen en het tijdstip waarop 80% van het totaal aantal gepote aardappelen gekiemd is, i.e. als de eerste blaadjes boven de grond zijn.

2.2.18 TotalStressRateModule

Deze module heeft de functie om de mate van groeireductie te berekenen voor elk plantorgaan dat een tekort aan water, stikstof of fosfor heeft. De groeireductie kan ook afhankelijk zijn van een opgelopen plantenziekte.

2.2.19 CropReaModule

Deze module bevat rekenregels om de gerealiseerde gewasgroei vanaf de opkomst te genereren. Er wordt rekening gehouden met water en/of stikstof tekort.

2.2.20 PostharversteModule

In deze module wordt het effect van het type bewaring, de omgevingstemperatuur bij bewaring, en de kwaliteit van de te bewaren aardappelen doorgerekend in termen van afname van de hoeveelheid versgewicht en van de hoeveelheid zetmeel.

2.2.21 NdemReaModule

In NdemReaModule wordt de gerealiseerde stikstofopname en –verdeling over de organen berekend die het aardappelgewas bij de actuele groei nodig heeft.

2.2.22 DoFinalCalcsModule

Berekende data vanuit andere modules wordt in deze module verzameld en omgerekend. Deze module voert een vooraf opgegeven set van gegenereerde gegevens uit naar de resultatenfiles.

3. Groei en ontwikkeling van het aardappel gewas

Don Jansen & Jacques Davies

3.1 Introductie

De snelheid waarmee drogestof wordt opgebouwd is afhankelijk van straling, temperatuur, beschikbaarheid van water en nutriënten en gewaskenmerken. Wanneer geen factoren de opbrengst reduceren of limiteren is de gewasgroei evenredig met de hoeveelheid geabsorbeerde fotosyntheseactieve straling (Haxeltine & Prentice, 1996, Dewar et al., 1998). De verhouding tussen de nettoproductie en de geabsorbeerde fotosyntheseactieve straling wordt de gebruiksefficiëntiefactor of ook wel Light Use Efficiency (LUE) genoemd. Vaak wordt aangenomen dat de LUE in termen van droge stof productie voor een bepaald gewastype of cultivar constant is, zoals in de LINTUL-type modellen (Spitters & Schapendonk, 1990; Ittersum et al., 2003). In TIPSTAR wordt de LUE in termen van suikerproductie echter afhankelijk gesteld van CO_2 en O_2 concentraties en dagelijkse temperatuur en stralingsintensiteit zoals beschreven in Rodriguez et al. (1999). De geproduceerde suiker wordt vervolgens verdeeld over bladeren, stengels en knollen. Hierbij wordt géén vaste of ontwikkelingsstadium afhankelijke verdeling over de verschillende plantorganen gebruikt, maar is aangenomen dat de geproduceerde suikers eerst naar de bladeren en stengels gaan en dat knollen pas gevuld worden als er een overschot aan suikers is (conform Kraalingen, pers. comm.). De hoeveelheid suiker die nodig is voor bladgroei is gerelateerd aan de mogelijke toename van het bladoppervlak. Bij lage LAI is deze exponentieel, gemodificeerd met een effect van temperatuur, terwijl bij hogere LAI een lineaire, temperatuurgemodificeerde, toename van het bladoppervlak mogelijk is zolang de temperatuursom van het gewas beneden een bepaald maximum is. Met behulp van het specifieke bladoppervlak (Specific Leaf Area, SLA) wordt de groei van bladoppervlak omgerekend naar groei van droge stof. De groei van droge stof wordt daarna omgekend naar een hoeveelheid suikers via een gemiddelde behoefte aan suikers per kg geproduceerde droge stof in blad (GVlleaves, voor achtergrond zie Penning de Vries et al., 1989).

In het vervolg van dit hoofdstuk wordt ingegaan op de manier waarop voor situaties waarin water en stikstof niet als mogelijk beperkende factoren worden meegenomen. In volgende hoofdstukken wordt besproken hoe effecten berekend worden van een beperkte beschikbaarheid van water en stikstof.

3.2 Beschrijving basis module voor gewasgroei

3.2.1 Fotosynthese

De drogestof productie in het gewas is evenredig met de hoeveelheid opgevangen straling door het gewas. Gebleken is dat de verhouding tussen de drogestof productie en de hoeveelheid geabsorbeerde fotosyntheseactieve straling onder verschillende omgevingsfactoren en omstandigheden constant is. De potentiële koolstofproductie SourcePot, (kg $DS.ha^{-1}.d^{-1}$) is evenredig met de hoeveelheid straling PAR, ($J.m^{-2}.d^{-1}$) en de bodembedekking FINT ($m_{blad}^2.m_{bodem}^2$):

SourcePot = PAR * FINT * LUE_Oijen * 10. * 1.e-6

met

PAR = PARFraction * RDD
FINT = 1.- exp (-ECPDFReal * LAITot)

LUE_Oijen is Light Use Efficiency (g $CH_2O.MJ^{-1}$) berekend zoals beschreven in Rodriguez et al. (1999), PARFraction is de fractie straling welk fotosyntheseactief is (dimensieloos), RDD is de dagelijkse globale straling ($J.m^{-2}.d^{-1}$), LAI is

groen bladoppervlakte index ($m_{blad}^2.m_{bodem}^{-2}$) en ECPDFReal is de geschatte extinctie coëfficiënt ($ha_{bodem}.ha_{blad}^{-1}$). De factoren 10 en 1.e-6 betreffen respectievelijk de omrekening van g m^{-2} naar kg ha^{-1} en J naar MJ.

3.2.2 Ontwikkeling blad en stengel

3.2.2.1 Bladoppervlak

De toename in bladoppervlak per plant kan beschreven verdeeld worden in twee delen: het exponentiële en het lineaire groeiproces.

```
LeafArGro;wthPl1 = LeafArPl * (exp (max(-30.,min(30.,RGRL*TmEff*Delt)))-1.)
LeafAr;GrowthPl2 = LeafArGrowthRef * Tmeff
```

LeafArGrowthPl1 is de exponentiële bladoppervlakte toename ($m_{blad}^2.plant^{-1}.d^{-1}$), LeafArGrowthPl2 is de lineaire leaf bladoppervlakte toename ($m_{blad}^2.plant^{-1}.d^{-1}$), LeafArPl is het totaal bladoppervlak per plant ($m_{blad}^2.plant^{-1}.d^{-1}$), RGRL is de relatieve bladgroeisnelheid tijdens exponentiële groeiproces $(°C.d)^{-1}$, Delt is de integratie tijdstap (d), LeafArGrowthRef is de lineaire bladgroeisnelheid ($m_{leaf}^2.plant^{-1}.\ °\ C^{-1}.d^{-1}$). De dag-effectieve temperatuur,TMEff (° C), wordt berekend als het verschil tussen de dagelijkse gemiddelde temperatuur,TMDA (°C) en een specifieke basis-temperatuur TmBase (°C) waaronder geen bladgroei mogelijk is. Tevens wordt een negatieve waarde van TMEff uitgesloten.

```
TMEff = MAX (TMDA-TmBase,0.)
```

De 'mogelijke' groei in bladoppervlak per plant (LeafArGrowthPlP, $m_{blad}^2.plant^{-1}.d^{-1}$) is het minimum van de potentiële groei bij het exponentiële en bij het lineaire groeiproces, vermenigvuldigd met een reductiefactor (StressIndexLvAr; dimensieloos):

```
LeafArGrowthPlP = StressIndexLvAr * min (LeafArGrowthPl1, LeafArGrowthPl2)
```

Als de opbrengstlimiterende factor stikstof niet wordt gesimuleerd, wordt reductiefactor StressIndexLvAr bepaald door de 'leeftijd' van het gewas, uitgedrukt in een temperatuursom (TMSUM, °C.d) ten opzichte van een cultivar specifieke maximale temperatuursom (TMSumLeafGrowth °C.d) waarbinnen die cultivar nog blad kan aanmaken. Bij nadering van de temperatuursom van het gewas aan die maximale temperatuursom zal de snelheid van bladvorming afnemen naar 0:

```
StressIndexLvAr = max(0.,(1.-(TMSum/max(1.e-6, TMSumLeafGrowth))))**LeafPar
```

LeafPar (dimensieloos) is hierin een cultivar specifieke parameter die de relatieve snelheid van afname van de groei van bladoppervlak weergeeft.
De berekening van deze StressIndexLvAr vindt plaats in submodel TotalStressStateModule.

De temperatuursom, TMSum wordt berekenend door vanaf opkomst van elke dag de dag-effectieve temperatuur te bepalen en in de tijd te integreren. Als water- en/of stikstoftekort stress veroorzaken, zou dit een effect kunnen hebben op de fysiologische betekenis van de temperatuursom. Om dit mogelijk te maken is een stressfactor StressIndexTmSu in de berekening opgenomen. Momenteel is StressIndexTmSu niet geëffectueerd en heeft daarom continue de waarde 1:

```
TMSum = intgrl (TMSum, TMEff * StressIndexTmSu, Delt)
```

3.2.2.2 Gewicht van blad en stengel

De mogelijke groei van het bladoppervlak per plant wordt omgezet naar mogelijke totale groei van bladopper van het gewas, LeafArGrowthP, ($m_{blad}^2.ha_{bodem}^{-2}.d^{-1}$), door vermenigvuldiging met de plantdichtheid NPL, (aantal planten.ha^{-1}):

```
LeafArGrowthP    = LeafArGrowthPlP * NPL
```

Toename van het bladgewicht per oppervlak, LeafWtGrowth, (kg_{blad} DS.$ha^{-1}.d^{-1}$) wordt verkregen door deling van de LeafArGrowth door de specifieke blad oppervlakte SLA, ($ha_{blad}.kg_{blad}^{-1}$):

```
LeafWtGrowthP   = (LeafArGrowthP / 10000.) / SLA
```

De factor 10000 betreft de omrekening van ha naar m^2.

De toename van het stengelgewicht wordt mede bepaald door het product van een fractiefactor LeafStemRatio (dimensieloos) en de toename van het bladgewicht LeafWtGrowth:

```
StemWtGrowthP = LeafStemRatio * LeafWTGrowthP * StressIndexStWt
```

De reductiefactor StressIndexStWt kan gebruikt worden om eventuele effecten van stikstof- en/of watergebrek op de verhouding tussen stengel en blad gewichten te kwantificeren. Momenteel is deze reductiefactor niet geëffectueerd.

Naast stengelgewicht wordt ook een schatting gemaakt van de groei van het stengelvolume, dat gebruikt wordt bij de berekening van de N behoefte (§ 5.1.2):

```
StemVolGrowthP = StVolLfAreaRatio * (LeafArGrowthP / 10000.)
```

Hierin is STVolLfAreaRatio een schatting van de cultivar specifieke verhouding tussen stengelvolume en bladoppervlak.

Op basis van de specifieke behoefte aan glucose per eenheid droge stof in de verschillende organen (Penning de Vries et al., 1989), kan de totale behoefte aan glucose uitgerekend die gepaard zou gaan met het realiseren van de 'mogelijke' groei van stengel en blad:

```
AboveGlucoseUseP = LeafWtGrowthP * GVlLeaves + StemWtGrowthP * GVlStems
```

GVlLeaves en GVlStems zijn respectievelijk de behoefte aan glucose voor groei van blad en stengel. Dit zijn parameters die cultivar afhankelijk kunnen zijn.

In een situatie waarin de vraag aan glucose voor bovengrondse groei kleiner of gelijk is aan wat de plant via fotosynthese beschikbaar krijgt, zal de bovengrondse mogelijke groei ook gerealiseerd kunnen worden. Als er minder glucose via fotosynthese beschikbaar komt, is het mogelijk dat additionele glucose beschikbaar komt uit herverdeling van afstervende biomassa (eerste mogelijkheid) of ook uit gebruik van reserves in de aanwezige organen (tweede mogelijkheid):

```
if (SourcePotP < AboveGlucoseUseP) then
! eerst hergebruik van droge stof in dode organen: fractie redistributie van afstervende delen:
  FrRedistrUseP    = min(1., max(0., (AboveGlucoseUseP - SourcePotP ) /
                          (RedistrDMTot * GVlReserves * ConvReserveDM)))
  if (SourcePotP + FrRedistrUseP * ConvReserveDM * RedistrDMTot * GVlReserves <
    AboveGlucoseUseP) then
! als dat nog niet genoeg is: fractie gebruik van reserves in bestaande levende organen bepalen
    FrReserveUseP    = min(1., max(0., (AboveGlucoseUseP - SourcePotP –
                            FrRedistrUseP * ConvReserveDM * RedistrDMTot *
```

```
                                GVlReserves) /(ReserveDMTot*GVlReserves*ConvReserveDM)))
    end if
    LeafArGrowthPlP = max(0,LeafArGrowthPlP * (SourcePotP  + GVlReserves  * (
          (FrRedistrUseP * RedistrDMTot) +  (FrReserveUseP * ReserveDMTot) ) /
          AboveGlucoseUseP)
```

De hoeveelheid droge stof die beschikbaar is voor herverdeling uit stervend materiaal is het totaal van wat er in alle afstervende delen zit, en zo worden ook de beschikbare reserves in alle levende delen opgeteld tot een totaal:

```
RedistrDMTot = RedistrLeafDead + RedistrStemDead + RedistrTuberDead
ReserveDMTot = ReserveLeafLive + ReserveStemLive + ReserveTuberLive
```

Het totaal aan droge stof die te herverdelen is van het orgaan blad komt uit alle afstervende bladklassen (zie volgende paragraaf) en wordt opgeteld over alle afstervende bladklassen:

```
RedistrLeafDead = RedistrLeafDead + FreeDMleafCl(i)
```

FreeDMleafCl(i) is de hoeveelheid vrij beschikbare droge stof in klasse ‘i’, i.e. droge stof in de vorm van eiwitten en koolhydraten in enzymen en reserves en niet in structurele biomassa (celwanden, dna). Bij simulaties waarin niet de stikstof gelimiteerde groei wordt berekend, is de hoeveelheid vrij beschikbare droge stof in elk orgaan een vaststaande fractie van het levende gewicht. Deze fractie wordt aangegeven met de parameters frreserveleaf, frreservestem en frreservetuber, voor blad, stengel en knol.

In elke levende bladklasse wordt een deel van de daar in zittende vrij beschikbare droge stof beschikbaar gesteld en opgeteld over die levende bladklassen:

```
ReserveLeafLive = ReserveLeafLive + FreeDMleafCl(i) * kdestableaf
```

Hierin is kdestableaf de maximale fractie van de vrij beschikbare droge stof die per dag beschikbaar komt.

Op eenzelfde manier worden de herverdeling van doodgaande stengels en knollen en de beschikbare reserves uit de levende stengels en knollen berekend:

```
ReserveStemLive = (1. - StressIndexStAg) * FreeDMStemLive * kdestabstem
ReserveTuberLive = (1. - StressIndexTuAg) * FreeDMTuberLive * kdestabtuber
RedistrStemDead = StressIndexStAg * FreeDMStemLive
RedistrTuberDead = StressIndexTuAg * FreeDMTuberLive
```

StressIndexStAg en StressIndexTuAg zijn de dagelijkse gewichtsfracties van doodgaande stengels en knollen. In TIPSTAR wordt aangenomen dat er geen knollen afsterven, en heeft StressIndexTuAg dus de waarde 0, terwijl bij simulaties van situaties zonder stikstofbeperking de StressIndexStAg gelijk gesteld wordt aan de relatieve sterftesnelheid van blad:

```
FrDyingLeaves    = TotDeadLeafRate / LeafWt
StressIndexStAg  = FrDyingLeaves
```

Hierin is TotDeadLeafRate de totale dagelijkse snelheid van afsterven van bladmateriaal (kg ha^{-1} d^{-1}) en LeafWt de totaal aanwezige hoeveelheid bladbiomassa aan het begin van de dag (kg ha^{-1}).

3.2.2.3 Blad leeftijdsklassen

Bladeren aan een aardappelstengel vormen discrete eenheden. Een gemiddelde aardappelplant heeft aan de hoofdstengel 15 – 20 bladeren wanneer voldoende water en nutriënten aanwezig zijn. De meeste gewasgroeimodellen maken geen onderscheid tussen de verschillende leeftijdsklassen van de bladeren in het gewas. In TIPSTAR worden de verschillende leeftijdsklassen van de bladeren van de aardappelplant modelmatig gesimuleerd zodat er een betere benadering van het groeiproces wordt verkregen. Hierdoor kunnen de volgende processen beter worden beschreven:

1. Het ouder worden van bladeren
2. Het versneld ouder worden van bladeren die beschadigd zijn
3. Het versneld afsterven van bladeren door schaduw
4. Stikstof (re)distributie in bladeren
5. De groei van schimmels op de bladeren (zoals *Phytophthora infestans*) en het effect van deze schimmels op het afsterven en de fotosynthese van bladeren.

Bladleeftijdsklassen worden gevormd vanaf opkomst tot aan het einde van het groeiseizoen. In TIPSTAR wordt in elke tijdstap de toe- of afname berekend van elke bladleeftijdsklasse. De eerst gevormde bladleeftijdsklasse krijgt nummer 1, terwijl een nieuw gevormde leeftijdsklasse het hoogste nummer krijgt. Elke leeftijdsklasse heeft 5 karakteristieken:

1. Nummer leeftijdsklasse
2. Bladoppervlakte (m^2)
3. Blad (drogestof) gewicht (kg)
4. Blad leeftijd in termen van temperatuursom (°C. d)
5. Indicatie van levend, dood en aanwezig of dood en weggewaaid.

De karakteristieken van een nieuwe leeftijdsklasse worden bepaald gedurende de integratietijdstap:

```
RLeafArCl(LeafClCnt) = LeafArGrowth*delt
RLeafWtCl(LeafClCnt) = LeafWtGrowth*delt
LeafAgeCl(LeafClCnt) = 0.
LeafAliveCl(LeafClCnt) = .true.
```

LeafClCnt is de bladklasseteller, die begint bij 0 en voor elke dag dat er nieuw bladoppervlak gemaakt wordt met 1 opgehoogd wordt, RLeafArCl ($m^2.ha_{bodem}^{-1}$) het bladoppervlak en RLeafWtCl (kg DS. ha_{bodem}^{-1}) het bladgewicht van de nieuw gevormde bladklasse. Deze laatste twee worden verkregen door de groeisnelheden LeafArGrowth en LeafWtGrowth te vermenigvuldigen met de tijdstap Delt.; LeafAgeCL (°C. d) is de leeftijd van de bladklasse, terwijl LeafAliveCl de logische waarde “true” krijgt ter bevestiging van levend plantmateriaal. Als een klasse dood is krijgt LeafAliveCl de waarde “false’.

Als aanduiding van de fysiologische leeftijd van bladeren wordt een temperatuursom bepaald waarbij aangenomen is dat lagere lichtintensiteit een versnellende werking op de fysiologische veroudering heeft. Hiermee wordt de snellere veroudering (en daardoor eerder afsterven) van bladeren door beschaduwing onderin het gewas nagebootst.

```
RLeafAgeClP(i) = StressIndexLvAgClP(i) * (alphalvageT - betalvageT *
                                          PARTrans / 15.e6 ) * TmEff * delt
```

Waarin RLeafAgeClP(i) de snelheid van veroudering is in termen van verandering van temperatuursom voor bladklasse “i”, alphalvageT en betalvageT parameters, StressIndexLvAgeClP(i) een reductiefactor waarmee het effect van stikstof en/of watergebrek wordt berekend en PARTrans de geschatte lichtintensiteit is waaraan de bladklasse wordt blootgesteld. Zij wordt berekend door de extinctie van licht door jongere (=bovenliggende) bladklassen te bepalen:

```
PARTrans = PAR * exp (-ECPDFReal * LeafArSum / 10000.)
```

3.2.2.4 Afsterven van bladeren

Tijdens elke tijdstap wordt van elke leeftijdsklasse (i) de leeftijd van de bladklasse, LeafAgeCl(i) (ºC. d), vergeleken met de maximale leeftijd (LeafAgeMx; ºC. d). Wanneer dan de bladleeftijd van een bladleeftijdsklasse hoger ligt dan deze maximale leeftijd wordt verondersteld dat de gehele leeftijdsklasse in die tijdstap afsterft. Er zijn in het model daarmee 3 situaties waarin bladeren zich kunnen bevinden:
Bladeren die aan het afsterven zijn, een situatie die optreedt op de dag dat de LeafAgeCl(i) voor het eerst de maximum temperatuursom overschrijdt; uit deze bladeren kan nog droge stof onttrokken worden voor hergebruik elders in de plant.
Bladeren die afgestorven zijn, dit treedt op voor bladeren waarvan de temperatuursom minstens twee dagen de maximale leeftijd heeft overschreden. Van de droge stof in deze bladeren kan geen gebruik meer worden gemaakt voor hergebruik elders in de plant. Voorts zal elke dag een deel van deze bladeren afvallen.
Bladeren die niet zodanig verouderd zijn dat ze daaraan dood gaan; deze bladeren kunnen eventueel nog wel (deels) afsterven door ziekte. Verder kan een deel van de reserves in deze bladeren gebruikt worden voor processen elders in de plant.

Het totale gewicht van een bladklasse wordt verdeeld in 3 groepen: LeafWtLiveCl, LeafWtDeadCl en LeafWtLossCl. Voor bladoppervlak wordt hetzelfde gedaan (Wt wordt dan vervangen door Ar). De snelheden waarmee die groepen toe of afnemen worden berekend als volgt:

```
if ( (not(LeafAliveCl(i))) .and. (FreeDMLeafCl(i) > 0.)) then
! voor bladeren die aan het afsterven zijn: ze zijn dood, maar hebben nog vrije droge stof
   RLeafWtDeadCl(i) = LeafWtLiveCl(i) - FrRedistrUse * FreeDMLeafCl(i)
   RLeafWtLiveCl(i) = - LeafWtLiveCl(i)
   RLeafWtLossCl(i) = 0.
   RLeafArDeadCl(i) = LeafArLiveCl(i)
   RLeafArLiveCl(i) = - LeafArLiveCl(i)
   RLeafArLossCl(i) = 0.
elseif (not(LeafAliveCl(i))) then
! voor bladeren die minstens twee dagen dood zijn en geen vrije droge stof meer hebben
   RLeafWtLiveCl(i) = 0.
   RLeafWtDeadCl(i) = - LeafWtDeadCl(i) * RelLossDeadLeaves
   RLeafWtLossCl(i) =   LeafWtDeadCl(i) * RelLossDeadLeaves
   RLeafArLiveCl(i) = 0.
   RLeafArDeadCl(i) = - LeafArDeadCl(i) * RelLossDeadLeaves
   RLeafArLossCl(i) =   LeafArDeadCl(i) * RelLossDeadLeaves
else
! levende bladeren, waarin sterfte kan optreden door ziektes waarbij dan geen hergebruik
! van reserves en vrije droge stof optreedt
   RLeafWtDeadCl(i) =   (LeafWtLiveCl(i) - FrReserveUse * FreeDMLeafCl(i) *
                          kdestableaf) * StressIndexLvDyingCl(i)
   RLeafWtLiveCl(i) = - FrReserveUse * FreeDMLeafCl(i) * kdestableaf –
                        RLeafWtDeadCl(i)
   RLeafWtLossCl(i) = 0.
   RLeafArDeadCl(i) =   LeafArLiveCl(i) * StressIndexLvDyingCl(i)
   RLeafArLiveCl(i) = - RLeafArDeadCl(i)
   RLeafArLossCl(i) = 0.
end if
```

De nieuwe gewichten en oppervlakten per bladklasse worden verkregen door integratie van deze snelheden en de totalen van het gehele gewas door sommatie over alle bladklassen.

3.2.3 Wortelgroei

In TIPSTAR wordt de biomassa van wortels niet expliciet gesimuleerd. Belangrijke redenen hiervoor zijn dat er te weinig goede meetgegevens zijn om zinvolle aannames over de snelheid van groei te doen en dat de hoeveelheid biomassa in de wortels, zeker aan het eind van het groeiseizoen, waarschijnlijk te verwaarlozen is ten opzichte van de biomassa in de overige organen. Wel wordt in TIPSTAR de bewortelingsdiepte gesimuleerd, als belangrijk onderdeel in de dynamische processen betreffende opname van water en stikstof in de bodem. De bewortelingsdiepte bepaalt bij de berekening van de opnameprocessen of water en stikstof in bepaalde bodemlagen bereikbaar is voor de wortels.
In het model wordt aangenomen dat de maximale bewortelingsdiepte RootDepthMx (m) gelijk is aan de kleinste van de maximale bewortelingsdiepte van de aardappelplant RootDepthCropMx (m) en die van de betreffende bodem (RootDepthSoilMx, (m)).
Dieptegroei van wortels vindt alleen plaats wanneer er tevens bladgroei is en de dieptegroei is begrensd door een maximale groeisnelheid per dag, RootDepthGrowthPar ($m.d^{-1}$). Onder optimale omstandigheden, is de dagelijkse toename van de bewortelingsdiepte gelijk aan een maximale dieptegroeisnelheid (RootDepthGrowthPar, $m\ d^{-1}$), tenzij het verschil tussen de actuele worteldiepte RootDepth (m) en de maximale worteldiepte RootDepthMx kleiner is dan dit maximum:

```
RootDepthChangeP  = min (RootDepthGrowthPar, &
                         max (RootDepthCropMx - RootDepth, 0.)) * StressIndexRtDp
```

In situaties dat de watergelimiteerde productie wordt gesimuleerd, wordt de dieptegroei van wortels beperkt met een reductiefactor StressIndexRtDp.

3.2.4 Knolgroei

Zoals eerder vermeld, vindt groei van knollen alleen plaats wanneer er meer suikers via fotosynthese geproduceerd worden plus wat er uit reserves en herverdeling van afstervende biomassa beschikbaar komt dan dat er voor bovengrondse groei benodigd zijn:

```
AvailableGlucoseTuberP  = SourcePotP  + GVIReserves  *
                          ( FrRedistrUseP * RedistrDMTot  + FrReserveUseP * ReserveDMTot)

TuberWtGrowthP = max(0,  AvailableGlucoseTuberP - AboveGlucoseUseP ) / GVITubers
```

Hierin is GVITubers de hoeveelheid suiker die nodig is voor de productie van 1 kg droge stof in de knol ($kg\ CH_2O\ kg^{-1}$ ds).

Doordat de groei van de bladeren en daarmee van de stengels in de loop van het groeiseizoen minder wordt en uiteindelijk stopt, zal de relatieve hoeveelheid suikers die naar de knol gaat in de loop van het seizoen toenemen totdat uiteindelijk de knol alle netto geproduceerde suikers krijgt. Omdat na verloop van tijd bladeren zullen afsterven en er daardoor minder suikers voor groei beschikbaar komen, zal de absolute groei van de knol aan het einde van het groeiseizoen afnemen naar nul wanneer er geen levende bladeren meer aanwezig zijn.

4. Simulatie van watergelimiteerde groei en ontwikkeling

Don Jansen

4.1 Introductie

Als een plant groeit heeft zij water nodig, voor omzetting van chemische verbindingen, als medium voor verschillende van deze omzettingen en voor transport van anorganische en organische stoffen van de wortels naar de andere delen en andersom. Water, of beter gezegd, het verdampen van water door de plant is daarnaast belangrijk voor de koeling van de plant, met name voor de delen die in de zon staan, zoals de bladeren. Door instraling van de zon warmen deze delen op en zonder verdamping zouden zij zo warm kunnen worden dat fysiologische processen vertragen of zelfs geheel stoppen, bijvoorbeeld doordat eiwitten denaturaliseren.

De groei van een gewas kan beperkt worden door zowel een tekort als een teveel aan water. Bij een watertekort, zorgt de sterke binding van het resterende water aan de bodemdeeltjes voor een gereduceerde opname van water. Hierdoor zal de verdamping teruglopen, hetzij door een verminderd aanbod van water, hetzij doordat de huidmondjes sluiten. In veel gevallen resulteert dit in een reductie in de fotosynthese, hetzij als gevolg van het sluiten van de huidmondjes, hetzij door verminderde werking van enzymen door te hoge temperatuur, of door een combinatie van beide. Ook zal in het algemeen de onderhoudsrespiratie van het gewas teruglopen, waardoor onder gelijkblijvende weersomstandigheden er vaak een rechtlijnig verband wordt gevonden tussen hoeveelheid water dat door de plant verdampt wordt en de droge stof productie (e.g. de Wit, 1958).
Als een bodem zich vult met water, verminderd de hoeveelheid zuurstof in de bodem. Bij lage zuurstofbeschikbaarheid in de bodem, kunnen de wortels fysiologische processen waarbij zuurstof nodig is niet uitvoeren (zie ook Jackson & Drew, 1984). Als gevolg van een zuurstoftekort kan het voorkomen dat de wortels zodanig slecht functioneren of zelfs afsterven zodat de fotosynthese niet meer goed of geheel niet meer mogelijk is.

Om de dynamiek van het bodemwater te simuleren is een groot aantal simulatiemodellen ontwikkeld. Twee ervan kunnen momenteel in TIPSTAR gebruikt worden om te combineren met de gewasgroei- en bodem-organische stof modules:

- SAHEL (Van Keulen, 1975; Penning de Vries et al., 1989; Ridder & van Keulen, 1995), dat geschikt is voor bodems met diep grondwater waarin géén nalevering plaatsvindt vanuit het grondwater naar het water in het bewortelde deel van de bodem.
- SAWAH (Penning de Vries et al., 1989; Berge et al., 1992; Wopereis et al., 1994; Berge et al., 1995), waarmee ook die nalevering gesimuleerd wordt en dat geschikt is voor bodems met (tijdelijk) waterverzadigde delen van het bodemprofiel, bijv. door hoog grondwater of waterondoorlaatbare lagen in het profiel.

Deze modellen zijn in de opgegeven referenties uitgebreid beschreven en er is geen noodzaak om dat in dit rapport te herhalen. Wel is van belang om de communicatie tussen de bodem water balans modellen en andere simulatie modulen te beschrijven.
De parameterisatie van de bodem water balans module (SoilWatModule) vindt plaats via de SoilWatDataModule. Deze leest een datafile of database uit en extraheert de benodigde gegevens voor het gekozen bodemtype. Deze gegevens worden via gestandaardiseerde namen op het schoolbord doorgegeven aan de waterbalans module en aan eventuele andere modules die deze informatie nodig hebben. De in de SoilWatModule uitgerekende vochtgehaltes van de diverse onderscheiden bodemlaagjes worden via het schoolbord doorgegeven aan o.a. de WaterUptModule, waarin de dagelijkse snelheid van opname van water per bodemlaagje door het gewas wordt uitgerekend (zie onder). Deze opnamesnelheden worden op hun beurt weer doorgegeven aan de SoilWatDataModule om het vochtgehalte in de bodemlaagjes voor de volgende tijdstap uit te rekenen. De WaterUptModule is onafhankelijk van

het gekozen bodem water balans model, en één en dezelfde WaterUptModule kan gebruikt worden voor beide bodem water balans modellen.

4.2 Wateropname uit de bodem: WaterUptModule

Op basis van weersgegevens en het bladoppervlak wordt een potentiële dagelijkse transpiratiesnelheid van het gewas uitgerekend in EvapModule. Diverse berekeningsmethoden voor de relatie tussen weer, gewas en potentiële transpiratie zijn beschikbaar, bijvoorbeeld die volgens Makkink en volgens Penman (zie Kraalingen & Stol, 1997). Deze potentiële transpiratie (xTranspot; mm d^{-1}) wordt omgerekend naar een potentiële opname per doorworteld bodemlaagje (xTrwl; mm d^{-1}), waarbij aangenomen is dat de relatieve effectiviteit van opname van water door wortels (RootEff) lager wordt naarmate de bodemlaag dieper is. Hiermee wordt het patroon nagebootst van beworteling waarbij relatief minder lengte en volume van wortels in diepere lagen gevonden wordt:

```
RootEff(i) = 1./(1.+exp(RootEffAlpha * (0.5*(LLimit(i) + ULimit(i)) – RootEffBeta)))
Trwl(i) = Transpot * max(0., min(1., zrtl(i) * RootEff(i) / ZrtT))
```

met: RootEffAlpha en RootEffBeta parameters waarmee de afname van de effectiviteit van wateropname in de diepte van de bodem wordt bepaald; LLimit en ULimit (m) de diepte van de ondergrens en bovengrens van het bodemlaagje t.o.v. het bodemoppervlak; zrtl het doorwortelde deel van het bodemlaagje (m) en ZrtT de som van zrtl * RootEff over alle bodemlaagjes.

Als een bodemlaagje uitdroogt, neemt de beschikbare hoeveelheid water voor opname door de plant af. In termen van de relatieve en absolute hoeveelheid water die in een bodemlaagje beschikbaar is (Wcrel en WaterExtrLayerMx, mm d^{-1}) wordt dit berekend via:

```
Wcrel(i) = min(1., max(0., (Wclqt(i) – Wcwp(i)) / (Wcwu(i) – Wcwp(i))))
WaterExtrLayerMx(i) = min( max(0., wclqt(i) - wcwp(i)) * tkl(i),
                                    WaterExtrPar * zrtl(i) * wcrel(i)) * 1000
```

waarin Wclqt = de actuele volumetrische hoeveelheid water in een bodemlaatje (cm^3 water cm^{-3} bodem), Wcwp = de volumetrische hoeveelheid water (cm^3 water cm^{-3} bodem) in een bodemlaagje waarbij de plant geen water meer kan opnemen, i.e. bij een pF van 4.2; Wcwu = de volumetrische hoeveelheid water (cm^3 water cm^{-3} bodem) waaronder het gewas droogte stress begint te ondervinden; tkl = dikte van het bodemlaagje (cm); WaterExtrPar = maximale hoeveelheid water die per doorwortelde m bodem opgenomen kan worden (m m^{-1} d^{-1}); 1000 = omrekeningsfactor (mm m^{-1}).

In gevallen waarin de potentiële opname Trwl in alle bodemlaagjes gelijk of kleiner is aan de maximaal beschikbare hoeveelheid, kan de bodem voldoende vocht aanleveren en is er geen vochtgebrek. De gerealiseerde opname van water uit elk bodemlaagje is dan gelijk aan Trwl en de som daarvan over alle bodemlaagjes (Transact) is dan gelijk aan de totale potentiële verdamping Transpot.

Als in alle doorwortelde bodemlaagjes de maximaal beschikbare hoeveelheid WaterExtrLayerMx lager is dan de potentiële opname Trwl, is er in alle laagjes dus watergebrek, en wordt de gerealiseerde opname gelijk gesteld aan WaterExtrLayerMx. De Transact als som daarvan over alle laagjes is dan dus minder dan de totale potentiële verdamping Transpot.

Als in een deel van de doorwortelde bodemlaagjes vochtgebrek optreedt (i.e. WaterExtrLayerMx < Trwl) en in een ander deel niet (i.e. WaterExtrLayerMx $\geq$ Trwl), dan wordt aangenomen dat de wortels uit de laagjes zonder watergebrek (deels) het watergebrek uit de andere laagjes kunnen compenseren. Daartoe wordt voor zover mogelijk het 'tekort' aan beschikbaar water uit de 'te droge' laagjes verdeeld over de 'vochtige' laagjes. Indien in die 'vochtige' laagjes nog genoeg water beschikbaar is om het gehele tekort uit de 'te droge' laagjes te compenseren, zal het gewas geen vochtgebrek ondervinden (Transact = Transpot). Is er echter te weinig 'extra' water in de vochtige

laagjes om het tekort te compenseren, dan zal de gerealiseerde verdamping minder zijn dan de potentiële (Transact < Transpot).

4.3 Effecten van watergebrek

4.3.1 Indicatie van vochtgebrek

Als maat voor het vochtgebrek dat een gewas ondervindt, wordt de verhouding tussen actuele en potentiële verdamping genomen, die wordt uitgerekend in TotalStressRateModule:

WaterStressIndex = Transact / Transpot

Er wordt in TIPSTAR geen cumulatief vochtgebrek bijgehouden, waarmee impliciet aangenomen is dat alle effecten van droogte of wateroverlast momentaan zijn. Dit houdt in dat berekeningen van effecten van droogte op groei en ontwikkeling alleen gedaan worden in de TotalStressRateModule en niet in de TotalStressStateModule. Alleen het effect op dieptegroei van wortels is afhankelijk van de status van het bodemvocht en wordt daarom in de TotalStressStateModule berekend.

4.3.2 Effecten op groei en ontwikkeling bovengronds

4.3.2.1 Suikerproductie

In de CropReaModule wordt het effect van watergebrek of overschot op de productie van suikers berekend door een stressfactor te vermenigvuldigen met de in de CropPosModule berekende mogelijke suikerproductie:

SourcePot = SourcePotP * StressIndexLue

Indien alleen effecten van watergebrek of overschot op de suikerproductie berekend worden, is StressIndexLue gelijk aan de WaterStressIndexLue, die berekend wordt via een Michaelis-Mente curve op basis van de WaterStressIndex:, en waarden heeft tussen 0 (bij hoge stress) en 1 (bij afwezigheid van stress):

WaterStressIndexLUE = (WaterStressIndex *(1. + alphaWater)) / (WaterStressIndex + alphaWater)

Hierin is alphaWater een (mogelijk cultivarafhankelijke) parameter.

4.3.2.2 Ontwikkeling bladoppervlak

Naast een effect op de suikerproductie, is het ook te verwachten dat bij droogte en wateroverschot relatief minder blad (en stengel) wordt aangelegd. Hiertoe wordt in CropReaModule de mogelijke groei van het bladoppervlak per plant (LeafArGrowthPlP) vermenigvuldigd met een stressfactor

LeafArGrowthPl = LeafArGrowthPlP * StressIndexLvAr

Wanneer alleen droogte en wateroverlast worden meegenomen, is deze StressIndexLvAr gelijk aan de WaterStressIndexLvAr, die berekend wordt volgens:

WaterStressIndexLvAr = 1. + 1. /(1.+exp(max(-30.,min(30., -KstrsAr*(WaterStressIndex - BetastrsAr)))) - 1. /(1.+exp(max(-30.,min(30., -KstrsAr*(1. - BetastrsAr)))))

Bij afwezigheid van waterstress zal deze vergelijking de waarde 1 krijgen, waarna bij toenemende waterstress eerst vrijwel geen effect op de relatieve groei van het bladoppervlak wordt gevonden en pas bij sterke stress een toename van het effect. KstrsAr en BetastrsAr zijn (mogelijk cultivar specifieke) parameters.

4.3.2.3 Veroudering blad

Bij tekort of overschot aan water is ook te verwachten dan aanwezige bladeren eerder zullen afsterven dan als er geen waterstress is. Hiertoe wordt in CropReaModule de mogelijke toename van de 'fysiologische leeftijd' van een bladlaagje (RLeafAgeClP) zoals berekend in CropPosModule vermenigvuldigd met een stressfactor StressIndexLvAgCl:

```
RLeafAgeCl(i) = RLeafAgeClP(i) * StressIndexLvAgCl(i)
```

De StressIndexLvAgCl is voor alle bladlaagjes identiek
Wanneer alleen droogte en wateroverlast worden meegenomen, is deze StressIndexLvAg gelijk aan de WaterStressIndexLvAr, die berekend wordt volgens:

```
WaterStressIndexLvAg = 1. + Gage * exp(max(-30., min(30., -KstrsAge * WaterStressIndex)))
```

Bij toenemende waterstress zal deze stressfactor exponentieel groter worden waardoor bladeren sneller 'verouderen' en eerder zullen afsterven. In feite is daarmee een cumulatief effect van waterstress op het gewas meegenomen KstrsAge en Gage zijn (mogelijk cultivar specifieke) parameters..

4.3.3 Groei van wortels in de diepte

Wanneer het volumetrisch watergehalte in de diepste bodemlaag waarin zich wortels bevinden, wclqt(i), ($cm_{H2O}^3.cm_{bodem}^{-3}$), lager is dan het volumetrisch watergehalte waarbij de plant continue verwelkt is, wcwp(i) ($cm_{H2O}^3.cm_{bodem}^{-3}$)(eng: wiltingpoint), dan wordt de dieptegroei van wortels gestopt door de logische parameter RootGrowthFlag op .false. te zetten, hetgeen doorwerkt in de berekening van de stressfactor:

```
if (RootGrowthFlag) then
   WaterStressIndexRtDp = 1.
else
   WaterStressIndexRtDp = 0.
 end if
```

Wanneer alleen waterstress meegenomen wordt, is de StressIndexRtDp gelijk aan de WaterStressIndexRtDp en wordt de mogelijke wortelgroei:

```
RootDepthChangeP = min (RootDepthGrowthPar, max (RootDepthCropMx - RootDepth, 0. ) ) *
StressIndexRtDp
```

5. Simulatie van stikstofgelimiteerde groei en ontwikkeling

Don Jansen

5.1 Dynamiek van N in het gewas

5.1.1 Introductie

In TIPSTAR wordt aangenomen dat stikstof in 3 verschillende groepen voorkomt in elk van de verschillende plant organen:

- Structureel gebonden N (NStruc); i.e. N die vastzit in DNA, celwanden en andere componenten die niet hergebruikt kunnen worden. Per gevormde structurele eenheid in een specifiek plant orgaan, i.e. oppervlak voor bladeren, volume voor stengels, en droge stof gewicht voor knollen, is een orgaan (en mogelijk cultivar) specifieke hoeveelheid structureel gebonden N nodig.
- Stabiel gebonden N (NStab); i.e. N aanwezig in eiwitten, reserves en andere componenten die voor een deel wel hergebruikt kunnen worden. Per gevormde structurele eenheid is er een orgaan (en mogelijk cultivar) specifieke minimum hoeveelheid die in deze componenten moet zitten en een maximum dat daarin kan zitten.
- N opgelost (NSol) in vocht aanwezig in cel, floeem en xyleem.

Voorts wordt aangenomen dat alle N die door de plant uit de bodem opgenomen wordt allereerst in de NSol terecht komt. N die nodig is om nieuwe plantendelen te laten groeien komt uit de NSol en uit de N die vrijkomt uit afstervende plantendelen en uit de reserves in de plant. Per dag is slechts een deel van de NSol beschikbaar om omgezet te worden in NStruc of NStab.

Afhankelijk van de verhouding tussen 'vraag' en 'aanbod' van N wordt berekend of alle mogelijke groei ook gerealiseerd kan worden en welke hoeveelheid stabiel gebonden N in de organen zal komen.
De minimale 'vraag' naar N wordt berekend door de mogelijke groei per orgaan zoals uitgerekend in CropPosModule te vermenigvuldigen met het gehalte aan NStruc plus het minimum gehalte aan NStab, terwijl voor de maximale 'vraag' het gehalte aan NStruc plus het maximum gehalte aan NStab wordt aangehouden.

Als er niet voldoende N beschikbaar is om te voldoen aan de minimale vraag aan N, dan wordt in eerste instantie de groei van de knollen verminderd. Als zelfs bij een nulgroei van de knollen nog een N tekort is, dan pas wordt de groei gereduceerd van de bovengrondse delen (zo die er nog is). Deze berekening vindt plaats in NDemPosModule.

Is er meer N beschikbaar dan de minimale vraag, dan wordt zo veel mogelijk voldaan aan de maximale vraag en wordt de NStab in de groeiende delen hoogstens gelijk aan het maximum. De NSol die dan eventueel nog beschikbaar is blijft in oplosbare vorm.

In TIPSTAR is opname van N alleen mogelijk vanuit de bodem (dus het effect van bladbemesting wordt niet gesimuleerd). De C en N dynamiek in de bodem zoals beïnvloed door organische stofgehalte van de bodem en toevoegingen van organische en anorganische vormen van N (zoals drijfmest, inwerken residuen voorvrucht, ammonium-nitraat kunstmest) bepaalt hoeveel anorganische N in het bodemvocht ter beschikking komt voor opname door het gewas. De SOM (Soil Organic Matter) module, zoals beschreven in Verberne et al.. (1990), Verberne et al. (1995) en Jongschaap (1996) is momenteel in Tipstar de enige beschikbare invulling van SoilNutModule om te combineren met de gewasgroei en de bodemwater modules.

5.1.2 Maximale en minimale behoefte aan N voor groei

Op basis van de in CropPosModule berekende mogelijke groei van de onderscheiden organen, wordt een maximale en minimale behoefte aan N (RNTotGrowthMax en –Min, kg N ha^{-1} d^{-1})om die mogelijke groei te realiseren berekend:

RNTotGrowthMax = RNTotStabMax + RNTotStruc
RNTotGrowthMin = RNTotStabMin + RNTotStruc

Waarin RNTotStruc: de totale behoefte aan structurele N, RNTotStabMax en –Min: de totale behoefte aan stabiele N (alle in kg N ha^{-1} d^{-1}):

RNTotStabMax = RNStabLeafNewMax + RNStabStemNewMax + RNStabTuberNewMax
RNTotStabMin = RNStabLeafNewMin + RNStabStemNewMin + RNStabTuberNewMin
RNTotStruc = RNStrucLeafNew + RNSTrucStemNew + RNStrucTuberNew

De behoefte van elk orgaan (Leaf, Stem, Tuber) aan stabiel (maximum en minimum) en structureel N (kg N ha^{-1} d^{-1}) wordt berekend met de voor elk orgaan speciefieke mogelijke groei en cultivar en orgaanspecifieke parameters (NStabxxx Max en NStabxxxMin en NStrucxxxMin):

RNStabLeafNewMax = (LeafArGrowthP/10000.) * NStabLeafMax
RNStabStemNewMax = StemVolGrowthP * NStabStemMax
RNStabTuberNewMax = TuberWtGrowthP * NStabTuberMax

RNStabLeafNewMin = (LeafArGrowthP/10000.) * NstabLeafMin
RNStabStemNewMin = StemVolGrowthP * NStabStemMin
RNStabTuberNewMin = TuberWtGrowthP * NStabTuberMin

RNStrucLeafNew = (LeafArGrowthP/10000.) * NstrucLeafMin
RNStrucStemNew = StemVolGrowthP * NStrucStemMin
RNStrucTuberNew = TuberWtGrowthP * NStrucTuberMin

Met NStrucLeafMin, NStabLeafMax en – Min in kg N ha^{-1} blad; NStrucStemMin, NStabStemMax en –Min in kg N volume-eenheid^{-1} stengel en NStrucTuberMin, NStabTuberMax en –Min in kg N kg^{-1} ds.

In feite zijn NStrucStemMin, NStabStemMax en –Min ook te beschrijven als hoeveelheid N die naar de stengels gaat ten opzichte van de groei aan bladoppervlak, door StemVolGrowthP te vervangen door StVolLfAreaRatio * (LeafArGrowthP / 10000.) (§ 3.2.2.2):

RNStabStemNewMax = StVolLfAreaRatio * (LeafArGrowthP / 10000.) * NStabStemMax

Op dezelfde manier kunnen RNStabStemNewMin en RNStrucStemNew aan LeafArGrowthP gerelateerd worden.

Aangezien StVolLfAreaRatio in TIPSTAR als constante wordt beschouwd, zijn nieuwe constantes (voorbeeld alleen gegeven voor NStabStemMax) te definiëren met eenheid kg N in stengel ha^{-1} blad:

NStabStemMaxNew = StVolLfAreaRatio * NStabStemMax

Een schatting voor deze nieuwe parameters is te verkrijgen uit proefgegevens, hetgeen niet direct mogelijk is voor de parameters met die per volume-eenheid stengel gedefinieerd zijn, aangezien er over het algemeen geen meet-gegevens voor stengelvolume zijn en er in TIPSTAR geen éénduidige maat voor dit volume gegeven kan worden.

5.1.3 Hoeveelheid N beschikbaar voor groei

Naast de berekening van de behoefte aan N wordt er ook een inschatting gemaakt van de beschikbare hoeveelheid N (NAvailGrowth, kg N ha^{-1} d^{-1}):

NAvailGrowth = ksynstab * NSol + RedistrNtot + ReserveNtot + RNTuberResp

Waarin ksynstab = (mogelijk cultivarafhankelijke) fractie van NSol die dagelijks beschikbaar is voor opname in nieuw geproduceerde droge stof, RedistrNtot = hoeveelheid N die op die dag beschikbaar komt uit afstervende droge stof, ReserveNtot = fractie van de stabiele N die maximaal dagelijks beschikbaar is voor herverdeling, RNTuberResp = de hoeveelheid N die vrijkomt bij verademing van knollen.

Bij de berekening van de respiratie van de knollen is aangenomen dat in principe al het levende materiaal verademd kan worden met een maximum fractie per dag (RelRespTuber; d^{-1}), behalve de vrije reserves (FreeDMTuberLive; kg ds ha^{-1}), die al direct beschikbaar zijn voor herdistributie:

RNTuberResp = NStabTuberLive * StrucDMTuberLive * RelRespTuber / TuberWt

Waarin: NStabTuberLive: stabiel N in levende delen van de knollen (kg N ha^{-1}), StrucDMTuberLive = alle levende biomassa in de knollen behalve de vrije reserves (FreeDMTuberLive, kg ds ha^{-1}):

StrucDMTuberLive = TuberWt - FreeDMTuberLive
FreeDMTuberLive = TuberWt - DMNStrucTuber * NStrucTuberLive –
NStabTuberMin*TuberWt*DMNStabTuberMin

Met DMNStrucTuber: de minimale hoeveelheid droge stof die er per hoeveelheid structurele N in de knol moet zijn (kg ds kg^{-1} N); DMNStabTuberMin: de minimale hoeveelheid droge stof die er per kg N in de knol moet zijn (kg ds kg^{-1} N).

DMNStrucTuber en DMNStabTuberMin en hun gelijken voor blad (Leaf) en stengel (Stem) zijn geïntroduceerd om te zorgen dat bij afsterven van organen het gehalte aan N lager kan zijn dan de minimale hoeveelheid die er bij nieuw groeiend materiaal in moet zitten.

De N uit reserves en uit herdistributie van afstervend materiaal (kg N ha^{-1} d^{-1}) wordt per orgaan uitgerekend en gesommeerd tot de totalen:

ReserveNTot = ReserveLeafNLive + ReserveStemNLive + ReserveTuberNLive
RedistrNTot = RedistrLeafNDead + RedistrStemNDead + RedistrTuberNDead

Per bladlaagje wordt bepaald of het afsterft dan wel levend blijft. Van afstervende bladlaagjes wordt alle stabiele N beschikbaar gesteld voor herdistributie (RedistrLeafNDead; kg N ha^{-1} d^{-1}):

RedistrLeafNDead = Σ NStabLiveCl(i) (voor alle bladlaagjes i)

Van de levende bladlaagjes is een fractie (kdestableaf, d^{-1}) van het verschil tussen de aanwezige hoeveelheid stabiele N (NStabLiveCl; kg N ha^{-1}) en het minimum gehalte aan stabiele N beschikbaar voor groei:

ReserveLeafNLive = kdestableaf * Σ(NStabLiveCl(i) - NStabLeafMin * LeafArLiveCl(i)/10000.)

Op een vergelijkbare wijze wordt de redistributie van N uit afstervende stengels en knollen en de herverdeling van N uit de reservers in die organen berekend:

```
RedistrStemNDead  = StressIndexStAgP * NStabStemLive
ReserveStemNLive = (1 - StressIndexStAgP) * (NStabStemLive  - NStabStemMin  * StemVol) *
kdestabStem

RedistrTuberNDead = StressIndexTuAgP * NStabTuberLive
ReserveTuberNLive = (1 - StressIndexTuAgP) * (NStabTuberLive - NStabTuberMin * TuberWt -
RNTuberResp)  * kdestabTuber
```

5.1.4 Maximale mogelijke groeisnelheden bij N gebrek

Is de beschikbare hoeveelheid N (NAvailGrowth; kg N ha^{-1} d^{-1}) kleiner dan de minimale hoeveelheid N die nodig is om de mogelijke groei te realiseren (RNTotGrowthMin, kg N ha^{-1} d^{-1}), dan wordt in NDemPosModule de toegestane fractie van die mogelijke groei berekend. Bovengrondse delen (AllowedAboveGrowthNlim) krijgen daarbij voorrang op knollen (AllowedTuberGrowthNlim):

```
if (NAvailGrowth < RNTotGrowthMin .and. RNTotGrowthMin > 1.e-8) then
  NDemRest = RNStabLeafNewMin + RNStabStemNewMin +
                  RNStrucLeafNew + RNStrucStemNew
  if (NDemRest > 1.e-8) then
    if (NAvailGrowth <= NDemRest ) then
      AllowedAboveGrowthNlim = NAvailGrowth / NDemRest
      AllowedTuberGrowthNlim = 0.
    else
      AllowedAboveGrowthNlim = 1.
      AllowedTuberGrowthNlim = (NAvailGrowth - NdemRest) / (RNStabTuberNewMin +
                                        RNStrucTuberNew)
    end if
  else
    AllowedTuberGrowthNlim = max(0., NAvailGrowth / max(1.e-8, RNTotGrowthMin))
  endif
elseif (RNTotGrowthMin <= 1.e-8) then
  AllowedAboveGrowthNlim = 0.
  AllowedTuberGrowthNlim = 0.
Else
  AllowedAboveGrowthNlim = 1.
  AllowedTuberGrowthNlim = 1.
end if
```

Deze toegestane groei factoren worden doorgegeven aan NDemReaModule (zie § 5.3).

5.2 Stikstofopname uit de bodem

De opname door een gewas van N uit de bodem is sterk gerelateerd aan de wateropname door het gewas en de verdeling van wortels, stikstof en water in de bodem. Daarom is in TIPSTAR simulatie van N gelimiteerde groei alleen mogelijk in combinatie met die van watergelimiteerde groei. Benodigde modules zijn SoilWatModule, voor de berekening van het actuele watergehalte (WCLQT), SoilNutModule, voor de hoeveelheid in het bodemvocht opgeloste anorganische N (ANLay), WaterUptModule voor de opname van water door het gewas (TrLay) en NUptModule voor de opname van N (NUptL) door het gewas. Deze gegevens dienen per bodemlaag berekend en uitgewisseld te worden.

In NUptModule van TIPSTAR worden twee mogelijke routes voor opname van N onderscheiden: door diffusie en door massatransport met opname van water. Mogelijkheden voor beide routes worden berekend en vergeleken met elkaar en met wat er door het gewas opgenomen zou kunnen worden.

Als er geen maximum aan de N opname door het gewas zou worden gesteld, zou bij een voldoende groot aanbod in het model geen rem op de N opname komen en de hoeveelheid vrije N in het gewas (NSol) oneindig toe kunnen nemen. In eerste instantie was de concentratie aan vrije N in het gewas als remmende factor op de maximale opname gezet, maar dit resulteerde in sterk fluctuerende dagelijkse N opnames. Dit komt door de gebruikte tijdstap van 1 dag, waardoor bij een hoge NSol concentratie op dag t de opname van N op dag t+1 sterk gereduceerd werd, waardoor NSol op die dag laag werd waarna op dag t+2 weer een grote N opname mogelijk was, etc.. Om deze oneigenlijke fluctuaties te voorkomen, en toch nog een tijdstap van 1 dag aan te kunnen houden, is gekozen op de maximale dagelijkse N opname (NuptMaxP, kg N ha^{-1} d^{-1}) gelijk te stellen aan de maximale hoeveelheid N die nodig is om nieuwe groei mogelijk te maken (RNTotGrowthMax, § 5.1.2).

Bij de berekening van de opname die mogelijk door diffusie gebeurd moet bekeken worden hoeveel N in elk bodemlaagje beschikbaar is voor opname via diffusie (PNUpDif; kg N ha-1 d-1). Aangenomen is dat in principe per bodemlaag alle in het bodemvocht opgeloste anorganische N (ANLay) beschikbaar is voor de plant. Wordt er door de plant géén water uit een bodemlaag opgenomen wordt, dan kan er ook geen N via diffusie opgenomen worden en is de beschikbare fractie (NUpRed) van ANLay gelijk is aan 0. Wordt al het bodemwater in een bodemlaag opgenomen, dan is NUpRed gelijk aan 1 en kan in principe alle ANLay uit deze bodemlaag opgenomen worden. Aangenomen is dat de fractie beschikbare ANLay op een niet-lineaire manier gerelateerd is aan de opname van water als fractie van de totale hoeveelheid water in een laagje (FrTrans), waardoor de verhouding NUpRed / FrTrans hoger is dan 1 bij FrTans ongelijk aan 0 of 1:

```
FrTrans = TrLay(i) / (frvolumelayer(i) * TkL(i) * WCLQT(i)*1000.)
NUpRed = max(0, 1. - exp(-MaxNUpRateMax  * FrTrans))
PNUpDif(i) = ANLay(i) * NUpRed
```

Hierin is TrLay(i) de opname van water door het gewas uit bodemlaag i (mm d^{-1}); frvolumelayer(i) de fractie van het mogelijke volume dat door bodemlaag i wordt ingenomen (m3 m^{-3}), dit is 1 voor alle bodemlagen behalve van die lagen waarin de ruggen aangelegd zijn waarvoor dit getal tussen 0 en 1 ligt; TkL(i) de dikte (m) van laag i; WCLQT(i) de actuele volumetrische hoeveelheid water in laag i (m^3 water m^{-3} bodem); 1000 de omrekening van m naar mm (mm m^{-1}); MaxNUpRateMax een parameter voor het relatieve effect van FrTrans op de N opname (d).

De mogelijke opname van N via massatransport uit een bodemlaagje, wordt berekend door de hoeveelheid opgenomen water uit dat laagje te vermenigvuldigen met de concentratie aan opgeloste anorganische stikstof in dat laagje (NCLay; kg N cm^{-3} H_2O):

```
PNUpTran(i) = TrLay(i) * NCLay(i) * 1.e8 * 1.e-1
NCLay(i) = ANLay(i)/(frvolumelayer(i) * WCLQT(i) * TkL(i) * 1.e2 * 1.e8)
```

Waarin 1.e8: omrekening van ha naar cm^2; 1.e-1 omrekening van mm naar cm; 1.e2 omrekening van m naar cm.

Dan wordt gekeken wat de meest beperkende factor betreffende opname van stikstof is: de totaal mogelijke opname via diffusie (NDif; kg N ha^{-1} d^{-1}), via massa transport (NTran; kg N ha^{-1} d^{-1}) of de vraag van het gewas:

```
NDif  = Σ PNUpDif(i)
NTran  = Σ PNUpTran(i)
NDemT = max(min(NuptMaxP, max(NDif, NTran)), 0.)
```

Hierin is NDemT (kg N ha^{-1} d^{-1}) de totale vraag aan N die uit de bodem geleverd moet worden. Is de vraag vanuit de plant (NuptMaxP) minder is dan er vanuit de bodem aangeleverd kan worden (max(NDif, NTran)), dan hangt NDemT niet alleen van de bodemkarakteristieken af. Daarom wordt de NDemT verdeeld over de bodemlagen, eerst op basis van de wateropname per laagje (TrLay) als fractie van de totale wateropname door het gewas (TransAct). Wordt de hiermee berekende opname minder dan NDemt, dan wordt de resterende N die nodig is (NDemT-NUpTr ; waarin NUptTr de totale N opname uit de bodem is) over de bodemlagen verdeeld, waarbij de voorraad N in de bovenste bodemlagen als eerste aangesproken wordt:

De totaal beschikbare hoeveelheid anorganische N opgelost in het bodemvocht (kg ha^{-1}):

```
NAvail = Σ ANLay(i)
```

In een eerste stap wordt de opname van N verdeeld over de bodemlagen, proportioneel aan de fractie opgenomen water per bodemlaag:

```
Frac= TrLay(i)/TransAct
NUptL(i) = min(Frac*NDemT,ANLay(i))
```

De totale passieve opname via wateropname wordt dan berekend via optelling over alle bodemlagen:

```
NUpTr  = Σ NUptL(i)
NAvail  = NAvail - NUpTr
```

Indien de totale passieve opname niet voldoende is om aan de totale vraag te voldoen, wordt in een tweede stap uit bodemlagen waar water uit opgenomen wordt, de eventuele actieve opname bepaald, en wel volgens de aanname dat de opname van N preferentieel uit de bovenste bodemlagen plaatsvindt:

```
if ((NUpTr-NDemT < 0.) .and. (NAvail > 0.)) then
  do i=Istart,NL
    if (TrLay(i) > 0.) then
      NUptL(i)= NUptL(i) + min(ANLay(i)-NUptL(i), NDemT-NUpTr)
      NUpTr  = NUpTr  + min(ANLay(i)-NUptL(i), NDemT-NUpTr)
    endif
  enddo
endif
```

5.3 Effecten van stikstofgebrek op groei en ontwikkeling

In TIPSTAR worden effecten van mogelijk N gebrek op twee plekken berekend:

a. in TotalStressRateModule, waarin de effecten berekend worden van een lagere N opname dan benodigd om de mogelijke groei te kunnen realiseren; deze effecten worden gebruikt in CropReaModule.
b. in TotalStressStateModule, waarin de effecten worden berekend van de (te lage) N concentraties in het gewas op de groeiprocessen zoals gesimuleerd in CropPosModule

In TotalStressRateModule wordt de fracties toegestane groei, zoals berekend in NDemPosModule (§ 5.1.4), omgezet in stressfactoren die in de CropReaModule gebruikt worden om de gerealiseerde groeisnelheid uit mogelijke groeisnelheid te berekenen:

NitroStressIndexLvAr = AllowedAboveGrowthNLim
NitroStressIndexTuGr = AllowedTuberGrowthNLim

Hiermee is naast de relatieve verdeling van groei over de plant organen ook het effect van N gebrek op de absolute groei beschreven.

Ook eventueel watergebrek kan leiden tot een verandering in de verdeling van de mogelijke groei over bovengrondse delen en knollen. Aangenomen wordt dat het effect van watergebrek onafhankelijk is van dat van N gebrek. Daarom wordt het uiteindelijke effect van N en watergebrek op de relatieve groei van bovengrondse delen en knollen bepaald als het minimum van beide effecten. Dit houdt in dat de factor met het grootste effect op de groei bepaalt in welke mate de groei gereduceerd zal worden:

StressIndexLvAr = min (WaterStressIndexLvAr, NitroStressIndexLvAr)
StressIndexTuGr = min (WaterStressIndexTuGr, NitroStressIndexTuGr)

De berekening van het effect van watergebrek op de relatieve verdeling van droge stof naar de bladeren (WaterStressIndexLvAr) wordt beschreven in § 4.4.2.2. Aangezien in TIPSTAR het effect van watergebrek op de totale productie al wordt meegenomen, hoeft niet ook het effect ervan op de relatieve knolgroei te worden berekend., hetgeen inhoudt dat WaterStressIndexTuGr = 1.

In TotalStressStateModule wordt het effect van de N concentratie in het gewas berekend op:

a. LUE

Aangenomen is dat de LUE gereduceerd wordt als de concentratie van NStab in het blad afneemt:

NitrostressindexLUE = 1. - exp(-knlue * ((NStabconcLeafPhot / NStabLeafMax) - Noffset))

Hierin is NStabconcLeafPhot het gemiddelde van de NStab concentratie over alle bladlaagjes, gewogen naar de fractie opgevangen licht per bladlaagje; Noffset is de NStab concentratie relatief tot het maximum NStabLeafMax, waarbij de LUE = 0; knlue geeft de relatieve snelheid waarmee de LUE reduceert met afnemende NStab concentratie.

b. Ontwikkeling bladoppervlak

Aangenomen is dat de fractie suiker die naar de bovengrondse delen gaat relatief ten opzichte van de totale beschikbare hoeveelheid suikers gereduceerd wordt als de concentratie van NSol in het gewas afneemt:

NitroStressIndexLvAr = 1. - exp(-knrgrl*(NSolConc – bnrgrl))

Hierin is bnrgrl de NSol concentratie waarbij er geen suikers meer naar het blad zullen gaan; knrgrl geeft de relatieve snelheid waarmee de fractie suikers naar het blad reduceert met afnemende NSol concentratie.

c. Veroudering blad

Aangenomen is dat de veroudering van blad versneld naarmate de totale concentratie van N in het blad afneemt; dit wordt per bladlaagje berekend:

```
NitroStressIndexLvAgCl(i) = 1. + exp( -knlvag * (NtotConcLeafCl (i) - bnlvag))
```

Hierin is bnlvag de totale N concentratie in het blad waarbij de verouderingssnelheid is verdubbeld ten opzichte van die bij een optimale N concentratie ; knlvag geeft de relatieve snelheid waarmee de veroudering toeneemt met afnemende totale N concentratie.

d. Afsterven stengel

Om het meenemen van multiplicatieve effecten op sterfte door andere factoren (zoals Phytophthora) makkelijk te maken wordt eerst de fractie overlevende stengel brekend, waarbij aangenomen is dat deze afneemt met verlaging van de stabiele N concentratie in de stengel::

```
NitroStressIndexStAg = max(0., 1. - exp( -knstag * ((NStabConcStemVol / NStabStemMax) -bnstag)))
```

Hierin is bnstag de NStab concentratie in het blad als fractie van het maximale gehalte NStabStemMax waarbij de stengel totaal afsterft; knstag geeft de relatieve snelheid waarmee de afsterving toeneemt met afnemende NStab concentratie in de stengel.

Omdat in de TotalStressStateModule géén effect van watertekort wordt berekend (alle effecten worden in de TotalStressRateModule bepaald), worden de uiteindelijke stressfactoren zoals ze doorgegeven worden naar CropReaModule gelijk gesteld aan berekende N effecten:

```
StressIndexLUE  = NitroStressIndexLUE
StressIndexLvAr = NitroStressIndexLvAr
StressIndexStAg =  1. - NitroStressIndexStAg
StressIndexLvAgCl(i) = NitroStressIndexLvAgCl(i)
```

6. Modules voor Kennisoverdracht

Don Jansen

6.1 Introductie

Agrobiokon heeft onderzoek uit laten voeren naar belangrijke aspecten van de teelt van zetmeelaardappelen in Noord-Nederland. Een aantal van deze onderzoeken heeft geresulteerd in algemeen beschikbare toepassingen via het internet, zoals OptiRob (OptiRob, 2004), OptiRas (OptiRas, 2004) en OptiLoss (OptiLoss, 2004). Een aantal andere onderzoeken heeft meer inzicht gegeven over de effecten van weer en bodem op opkomstdag en opkomstfractie van zetmeelaardappelen. Om deze kennis te integreren in het Tipstar model (Jansen et al., 2003a; 2003b) zijn de gegevens geanalyseerd en omgezet in modules die door het simulatiemodel aangeroepen kunnen worden. Op deze wijze kunnen de effecten van managementkeuzes in de teelt en het bewaarsysteem van zetmeelaardappelen dynamisch meegenomen worden voor verdere analyses. In dit hoofdstuk worden de verschillende modules besproken.

6.2 Effect grondbewerking op bodemfysische eigenschappen

Wanneer de grond geploegd of gemengwoeld wordt, wordt in TIPSTAR aangenomen dat het doorploegde of doorwoelde deel van de bodem een fysieke en chemische samenstelling krijgt die het gemiddelde is van dat van de afzonderlijke, doorgewerkte, bodemlaagjes (in subroutine GetNewSoilProfile). Wanneer bodembewerking tot beneden de normale ploegvoor plaatsvindt, is in TIPSTAR voorts aangenomen dat er ook een verandering optreedt in de bodemfysische eigenschappen. Voor het doorploegde/doorwoelde deel worden dan de zgn. Van Genuchten parameters (die gebruikt worden in de bodemwater modules bij het berekenen van de waterdynamiek in de bodem) opnieuw geschat op basis van nieuwe (gemiddelde) organisch stof gehalte en korrelgrootte verdeling. Deze schatting, berekend via subroutine PedoTrans_VGN, volgt de procedures zoals beschreven door Wösten et al. (2001).

6.3 Organische stof dynamiek en bemesting

Voor het volgen van de organische stof dynamiek in de bodem wordt gebruik gemaakt van het 'Soil Organic Model' (SOM) zoals ontwikkeld door Verberne en anderen (Verberne et al., 1990, 1995). In dit model wordt in bodemlaagjes (met dezelfde dikte en fysieke kenmerken als die in de bodemwater-module) de hoeveelheid organische C en N bijgehouden. Indien bemest wordt met een organische substantie (dierlijke mest, compost, gemaaide groenbemester of vanggewas, etcetera), dan worden de C en N gehaltes (organisch en mineraal) aangepast tot en met de diepte waarop die organische substantie ingewerkt is. Bij kunstmest wordt alleen het minerale N gehalte aangepast. Op basis van het veranderde gehalte aan organische stof worden de bodemfysieke characteristieken van de bodemlaagjes opnieuw geschat met behulp van de subroutine PedoTrans_VGN (zie ook § 6.2).

6.4 Opkomstdag en opkomstfractie

6.4.1 Introductie

Pootgoedkwaliteit is een belangrijk knelpunt in de aardappelzetmeelteelt. Pootgoedkwaliteit beïnvloed de opkomstdag en de opkomstfractie en is daarmee bepalend voor de aardappelzetmeelproductie. In 2001 en 2004 is een groot aantal aspecten van de kwaliteit van het pootgoed op opkomst onderzocht (Wijnholds, 2002 en Wijnholds, 2004). De kwantitatieve informatie uit deze experimenten is gebruikt om het begrip "Pootgoedkwaliteit" te koppelen aan het Tipstar simulatiemodel. Dit is gebeurd via de modules EMERGENCE voor de berekening van de maximale opkomstfractie en de dag waarbij 80% van deze fractie opgekomen is.
De karakteristieken van het pootgoed die in de analyse betrokken zijn betreffen

1. Bewaar-omstandigheden
   a. Bewaarsysteem: Cel, Gaaskist, Houten Kisten, Kiembakjes, of anders (bijv. Kuil)
   b. Temperatuur controle: Droogwand, Mechanische Koeling, Mechanische Ventilatie, Natuurlijke Trek, Ruimte Ventilatie en anders (bijv. Zuig Ventilatie)
   c. Wijze van drogen: Natuurlijke Trek, Temperatuurregeling, Temperatuur en Relatieve Vochtigheid regeling
2. Ziekten e.d.: beschadiging, rot, aantasting door zilverschurft, fusarium, rhizoctonia en schurft
3. Physiologische Conditie: waardering, veroudering, kiemlengte, onderwatergewicht
4. Afkiemen: of pootgoed wel / niet afgekiemd is voor poten

6.4.2 Benadering

Eerst is een model gemaakt waarin effecten van karakteristieken van het pootgoed op de dynamiek van opkomst worden beschreven. De parameters van dit model zijn gefit door ze te calibreren op de waarnemingen van de proeven in 2001 en 2004. In deze calibratie werd de som geminimaliseerd van de gemaakte fout tussen waargenomen en berekende opkomst van alle tijdstippen, alle behandelingen en alle herhalingen:

$$SSQ = \sum_{t=1}^{n}\sum_{h=1}^{m}\sum_{p=1}^{k}\left(O_{t,h,p} - E_{t,h,p}\right)^2$$

Waarin:
t = waarnemingstijdstip; met n het totaal aantal tijdstippen
h = herhaling; met m het aantal herhalingen
p = pootgoed partij met specifieke kenmerken; met k het aantal partijen
O = observatie van fractie opkomst
E = met model berekende fractie opkomst

Tabel 1. Aantal waarnemingen per ras in 2001 en 2004. In 2004 zijn de gegevens van één partij van ras Aveka niet meegenomen in de calibratie.

Jaar en ras	Waarnemingstijdstippen	Herhalingen	Partijen	Totaal observaties
2001 Seresta	5	1	48	240
2004 Seresta	6	2	52	624
2004 Mercator	6	2	47	564
Totaal				1428

6.4.3 Model

Om de dynamiek van opkomst te berekenen is een Gompertz curve gebruikt:

Opkomst = MaxKiem * exp(exp(-AlphaKiem * ($TempSum_i$ - TempSumKiem))))

Waarin:
Opkomst = fractie opkomst op dag i (-)
MaxKiem = fractie maximale opkomst (-)
AlphaKiem = relatieve veranderingssnelheid van opkomst bij verandering van Temperatuursom ($graaddag^{-1}$)
$TempSum_i = \sum_{j=pootdag}^{i} (T_j - 5.)$; (graaddag) waarin T_j is de gemiddelde temperatuur op dag j
TempSumKiem = TempSum waar $Opkomst_i$ is 50% van MaxKiem

De temperatuursom waarbij 80% van de maximale opkomst bereikt wordt is dan te berekenen via

$TempSum_{80\%}$ = TempSumKiem - ln(-ln(0.8)) / AlphaKiem

In EMERGENCE wordt vanaf poten de dagelijkse temperatuursom $TempSum_i$ bijgehouden en de dag waarop deze voorbij de $TempSum_{80\%}$ komt wordt beschouwd als opkomstdatum.
De totale hoeveelheid opgekomen planten wordt berekend door het aantal gepote planten te vermenigvuldigen met MaxKiem. Vanaf opkomst datum berekent Tipstar de groei van het gewas met die totale hoeveelheid opgekomen planten.

De 3 Gompertz variabelen AlphaKiem, MaxKiem en TempSumKiem worden gerelateerd aan de waargenomen kenmerken via de volgende berekeningen, waarbij X1 – X28 parameters zijn en X te vervangen is door elk van de variabelen AlphaKiem, MaxKiem en TempSumKiem:

$X = X_1 * (1 + (1 + effAfkiemen\_TempCont\_X) * effTemperatuur\_Controle\_X) *$
$(1 + (1 + effAfkiemen\_Bewaar\_X) * effBewaar\_X) *$
$(1 + (1 + effAfkiemen\_Drogen\_X) * effDrogen\_X) *$
$(1 + (1 + effAfkiemen\_Ziekten\_X) * effZiekten\_X) *$
$(1 + (1 + effAfkiemen\_Conditie\_X) * effConditie\_X)$

Het effect van Afkiemen wordt bepaald via:

$effAfkiemen\_TempCont\_X = X_2 * IsAfkiem$
$effAfkiemen\_Bewaar\_X = X_3 * IsAfkiem$
$effAfkiemen\_Drogen\_X = X_4 * IsAfkiem$
$effAfkiemen\_Ziekten\_X = X_5 * IsAfkiem$
$effAfkiemen\_Conditie\_X = X_6 * IsAfkiem$

Hierin is IsAfkiem = 1 als afkiemen heeft plaatsgevonden; 0 als dat niet zo is geweest;

Het effect van de temperatuurcontrole wordt bepaald via:

$effTemp\_Controle\_X = (X_7 * IsTempCont\_NT + X_8 * IsTempCont\_MV +$
$X_9 * IsTempCont\_DW + X_{10} * IsTempCont\_RV +$
$X_{11} * IsTempCont\_MK)$

Hierin is voor de temperatuurcontrole:
IsTempCont_NT = 1 bij Natuurlijke Trek; 0 als anders
IsTempCont_MV = 1 bij Mechanische Ventilatie; 0 als anders
IsTempCont_DW = 1 bij Droge Wand; 0 als anders
IsTempCont_RV = 1 bij Ruimte Ventilatie; 0 als anders
IsTempCont_MK = 1 bij Mechanische Koeling; 0 als anders

Als geen van bovenstaande: dan alle op nul (bijv. bij Zuigventilatie)

Het effect van het bewaarsysteem wordt bepaald via:
$effBewaar\_Systeem\_X = (X_{12} * IsBewaar\_C + X_{13} * IsBewaar\_G + X_{14} * IsBewaar\_H + X_{15} * IsBewaar\_K)$

Waarin:
IsBewaar_C = 1 als bewaar systeem is Cel; 0 als anders
IsBewaar_G = 1 als bewaar systeem is Gaaskist; 0 als anders
IsBewaar_H = 1 als bewaar systeem is Houten kist; 0 als anders
IsBewaar_K = 1 als bewaar systeem is Kiembakjes; 0 als anders

Als geen van bovenstaande: dan alle op nul (bijv bij kuil)

Het effect van drogen wordt bepaald via:

$effDrogen\_X = (X_{16} * IsDrogen\_NT + X_{17} * IsDrogen\_TR + X_{18} * IsDrogen\_TV)$

Hierbij is voor wijze van drogen:
IsDrogen_NT = 1 bij Natuurlijke Trek; 0 als anders
IsDrogen_TR = 1 bij Temperatuur Regeling; 0 als anders
IsDrogen_TV = 1 bij Temperatuur en RV Regeling; 0 als anders

Als geen van bovenstaande: dan alle op nul

$effZiekten\_X = (X_{19} * Beschadiging + X_{20} * Fusarium + X_{21} * Rot + X_{22} * Rhizoctonia + X_{23} * Schurft + X_{24} * Zilverschurft)$

* Beschadiging = beschadigings index: schaal 0 - 50
* Zilverschurft = index
* Fusarium = fractie aangetaste knollen
* Rot = fractie rotte knollen
* Rhizoctonia = index
* Schurft = % schil bedekt met schurft, gemiddelde van partij pootgoed

$effConditie\_X = (X_{25} * Waardering + X_{26} * Veroudering + X_{27} * Kiemlengte + X_{28} * OWG)$

* waardering = algemene indruk van de partij pootgoed: 2 = zeer slecht, 10 = uitmuntend
* veroudering = inschatting van fysiologische ouderdom van de partij pootgoed: 2 = zeer slecht, 10 = uitmuntend
* kiemlengte = lengte van de kiemen
* owg = onderwatergewicht van het pootgoed voorafgaande aan poten

6.4.4 Resultaten

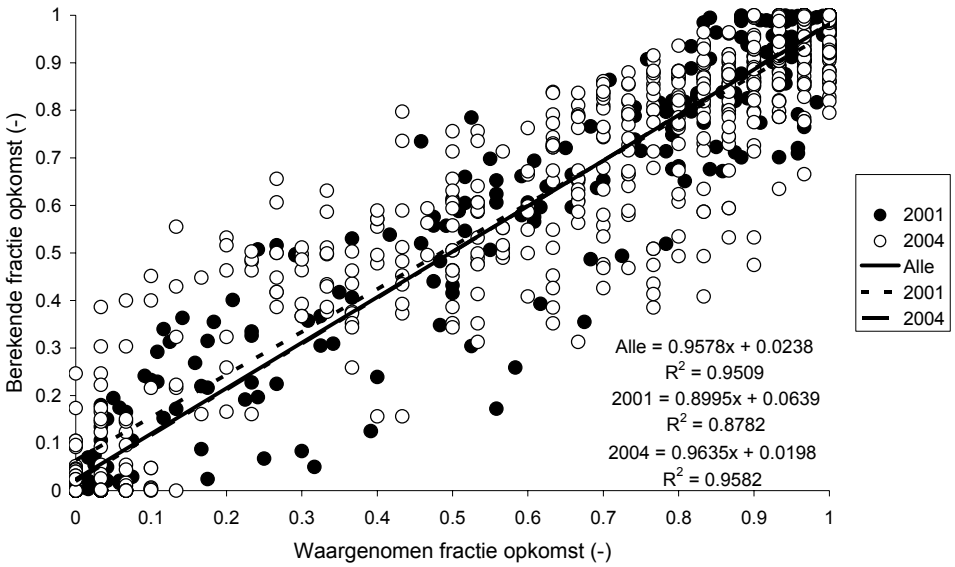
Tijdens de calibratie bleek dat er een jaareffect bestond dat niet aan de temperatuur vanaf poten kon worden toegewezen: in 2001 duurde het gemiddeld 34.4 graaddagen langer voordat de planten kiemden dan in 2004. Dit effect is mogelijk gerelateerd aan de condities, zoals temperatuur, waterbeschikbaarheid e.d., waaronder de aardappelen in beide jaren geproduceerd werden en/of waarbij ze bewaard werden. Met name bij bewaringsystemen waarbij de omgevingstemperatuur sterk de temperatuur in de opslag bepaald zou dit laatste een belangrijk deel van het effect kunnen bepalen.

Dit jaareffect is meegenomen in de calibratie door een extra parameter op te nemen bij de berekening van TempSumKiem: de parameter X_1 wordt vervangen door (X_1 + is2001*X_{29}), waarbij is2001 = 1 voor waarnemingen in 2001 en 0 voor waarnemingen in 2004.

Na calibratie blijkt dat een groot deel van de waargenomen opkomst gerelateerd kan worden aan de kwaliteitskenmerken (Figuur 2). De bijbehorende parameterwaarden staan in Tabel 2. Hierbij valt op dat parameter X_{22} de waarde 0 voor de 3 Gompertz variabelen. Dit houdt in dat er geen effect van Rhizoctonia gevonden is op de opkomstdynamiek. Dit kan met name te maken hebben aan het relatief lage verschil in partijen in Rhizoctonia besmetting, met name in 2004.

Tabel 2. Parameters na optimalisatie.

Param	MaxKiem	AlphaKiem	TempSumKiem
X1	1.0978	0.0580	128.8235
X2	-0.0474	-1.0016	-1.4026
X3	2.8921	-1.1740	2.0430
X4	-13.342	-1.9277	-1.8451
X5	1.0660	-1.2426	0.5363
X6	37.458	-1.1455	-0.4726
X7	0.00407	0.1915	-0.05555
X8	0.7682	-0.1206	-0.03311
X9	-0.001855	0.7017	-0.03663
X10	0.00417	0.19133	-0.00771
X11	0.3507	0.2394	-0.01582
X12	-0.00857	0.6719	-0.01850
X13	0.2360	0.3302	-0.03819
X14	0.02617	0.1289	-0.03115
X15	0.001829	0.4872	-0.1307
X16	0.02301	-0.1077	-0.06257
X17	0.04002	0.05586	-0.06151
X18	0.04514	-0.0947	-0.03856
X19	-0.00704	-0.00424	0.001627
X20	-2.7092	-2.6489	0.6477
X21	-0.8835	16.4639	0.07064
X22	0.	0.	0.
X23	0.	0.02490	0.001597
X24	0.000388	-0.00528	0.00003679
X25	-0.00902	-0.0001153	-0.002028
X26	0.01558	0.07508	0.02160
X27	0.00509	-0.06762	-0.01061
X28	-0.000104	-0.000900	0.0001132
X29	-	-	34.3705



Figuur 2. *Vergelijking tussen waargenomen en berekende fractie opkomst van beide jaren, alle waarnemingstijdstippen, herhalingen en rassen.*

6.5 Rooibeschadiging en bewaarverlies

Raymond Jongschaap & Don Jansen

6.5.1 Introductie

OptiRob, een Adviessysteem voor Rooibeschadiging en Bewaring van zetmeelaardappelen (OptiRob, 2004; http:\\optirob.kennisakker.nl) vormt de basis voor de module StockDamage (zie annex X). Door StockDamage wordt echter het bewaarverlies met een dynamisch temperatuursverloop berekend i.p.v. met een vaste temperatuur zoals in OptiRob. Zo kan het effect berekend worden van bewaring in kuil, sleufsilo en (geventileerde of een gekoelde) schuur onder variabel weer.

Korte (kuil) en lange bewaring (schuur) van zetmeelaardappelen zijn een belangrijk onderdeel van gewasmanagement. Tijdens de bewaring treden verliezen op die enerzijds onvermijdelijk zijn, zoals verliezen door rustademhaling, en anderzijds vermeden hadden kunnen worden zoals verliezen door onder andere rooibeschadiging, rot en ademhaling door te hoge of te lage temperaturen.

Bewaarverliezen zijn zeer variabel, moeilijk zichtbaar te maken en kunnen de teler tientallen tot honderden Euro's per hectare aan opbrengstderving kosten. Bewaarverliezen ten gevolge van rooibeschadiging zijn te voorkomen en leveren de teler direct een hoger rendement. Een correcte afstelling van de rooimachine voorkomt rooibeschadiging en leidt direct tot saldoverbetering.

De Tipstar module StockDamage maakt, net zoals OptiRob, bewaarverliezen inzichtelijk door onderscheid te maken tussen:

- onvermijdbare bewaarverliezen
- bewaarverliezen door rooibeschadiging
- bewaarverliezen door te hoge bewaartemperaturen (hoger dan 6 °C)

Verlies door koudeverzoeting bij te lage bewaartemperaturen (lager dan 6 °C) wordt nog niet berekend.

6.5.2 Onvermijdbaar bewaarverlies door rustademhaling

Bij een bewaartemperatuur tussen 5° en 6 °C zijn de knollen in rust. Tijdens deze rust ademen de knollen met een ademhalingssnelheid die zeer rasafhankelijk is. Deze verademing (Respiratie) is niet te beïnvloeden door externe omstandigheden. De bewaarverliezen veroorzaakt door deze verademing noemen we het onvermijdbare bewaarverlies (Bewaarverlies$_{R,}$ %). De waarden van het onvermijdbare bewaarverlies is rasafhankelijk en varieert tussen de 0.2% en 14.2% bij 6 maanden bewaring (Tabel 3).
In de module STOCKDAMAGE wordt het onvermijdbaar verlies van uitbetalingsgewicht (%) als functie van de rassenkeuze ingelezen. Deze lijst kan uitgebreid en aangepast worden. Bij een ras dat niet is opgenomen in de lijst (Tabel 3) wordt een gemiddelde waarde van 3.55% aangehouden.

Het onvermijdbaar bewaarverlies door verademing wordt op dagbasis uitgerekend. Daarvoor wordt een temperatuursom van 4 °C per dag voor een periode van 180 dagen (6 maanden) gesommeerd. Hierdoor is het mogelijk om voor kortere of langere bewaarperiodes alsmede bij veranderende bewaartemperaturen het onvermijdbaar verlies te berekenen als:

$$Bewaarverlies_R = \frac{R}{180} \cdot T$$

Met $Bewaarverlies_R$ = onvermijdbaar bewaarverlies door respiratie (% d^{-1}), T = gemiddelde dagtemperatuur (°C) en 180 = omrekeningsfactor van 6 maanden naar 1 dag.

Voor het gemiddelde bewaarverlies van 3.55% uitbetalingsgewicht per periode van 6 maanden, komt dat neer op een onvermijdbaar bewaarverlies door verademing van: 3.55%/(180 dagen· 4 °C)= 0.00493% $°C^{-1} \cdot d^{-1}$.

Tabel 3. Onvermijdbare bewaarverliezen van door verademing (respiratie) voor een aantal rassen voor een bewaarperiode van 6 maanden bij 4 °C (Brunt, 2004).

Ras	UBG[1] (%)	Drogestof (%)	Zetmeel (%)
Astarte	3.7	3.6	8.0
Elkana	2.0	3.8	10.1
Karakter	3.0	3.0	6.5
Kardent	2.8	2.1	6.5
Karnico	3.7	2.9	7.0
Kartel	2.8	2.9	6.9
Katinka	3.4	3.5	10.4
Nomade	3.1	3.4	10.8
Seresta	4.3	3.2	7.7
Festien	2.8	2.7	5.8
Goya	2.3	1.5	7.0
Ka91-1682	4.6	3.8	8.7
Aveka	2.5	2.6	6.1
Ku 91-088	1.9	2.3	8.9
Me 91 G-07	5.1	3.7	9.5
Me 93 f19	5.7	1.9	5.8
Mercator	5.8	4.9	10.2
Mercury	14.2	11.6	15.2
Plasinka	0.2	1.1	4.9
PR87-430	1.1	2.9	7.5
Starga	2.4	2.9	6.9
Stefano	0.6	1.1	5.0
Gemiddelde	*3.55*	*3.25*	*8.97*

[1] *UBG= Uitbetalingsgewicht.*

6.5.3 Vermijdbaar bewaarverlies door rooibeschadiging

Rooibeschadiging veroorzaakt wonden aan de huid en het vlees van de aardappel. De aardappelknol kan deze wonden helen. Bij het helingsproces, de eerste paar weken na het rooien, gaat de knol sneller zetmeel verademen waardoor er extra bewaarverliezen ontstaan. Bovendien verkurkt het gewonde vlees waardoor het OWG extra afneemt. De beschadigingindex (*BI*) geeft een indicatie voor de bewaarverliezen.

De beschadigingindex wordt vastgesteld door van een monster het aantal knollen (#) te tellen die geen, licht, matig en zware beschadigingen hebben. Deze klassen corresponderen met 0, 0-2%, 2-10% en >10% beschadigingen op het geschilde oppervlak (Figuur 3). De beschadigingindex *BI* (%) wordt berekend als:

$$BI = \frac{\left(3 \cdot Zwaar + 2 \cdot Matig + 1 \cdot Licht\right)}{6 \cdot MonsterGrootte} \cdot 100\%$$

Met *BI* = Beschadigings Index (%), *Zwaar* = het aantal knollen met >10% beschadigingen op het geschilde oppervlak, *Matig* = het aantal knollen met 2-10% beschadiging op het geschilde oppervlak, *Licht* = het aantal knollen met <2% beschadiging op het geschilde oppervlak, *Geen* = het aantal knollen zonder beschadiging op het geschilde oppervlak en *MonsterGrootte* = het totaal aantal knollen dat gebruikt is voor de analyse (= *Zwaar* + *Matig* + *Licht* + *Geen*).

Figuur 3. Zwaar (>10%), Matig (2-4%), Licht (<2%) en Geen (0%) beschadiging op het geschilde oppervlak.

In de module STOCKDAMAGE moet de waarde voor BI (0-50%) opgegeven worden. Het vermijdbaar bewaarverlies (Bewaarverlies$_{BI}$,% d^{-1}) wordt vervolgens berekend als:

$$Bewaarverlies_{BI} = \frac{\alpha \cdot BI^2 + \beta \cdot BI + \gamma}{180}$$

Met *BI* = Beschadigings Index (%), regressiefactoren *α, β en γ* (Tabel 4) en *180* = omrekeningsfactor van 6 maanden naar 1 dag. Bij een ras dat niet is opgenomen in de lijst worden de α, β en γ waarden gebruikt die voor *Alle* rassen gelden.

Tabel 4. Regressiefactoren α, β en γ per ras en voor alle rassen, voor de berekening van Bewaarverlies$_{BI}$.

Ras	α	β	γ
Astarte	0.0203	0.1225	3.7
Aveka	0.0203	0.1225	2.5
Festien	0.0119	-0.1041	2.8
Kantara	0.0124	-0.1932	2.5
Karakter	0.0000	-0.3231	1.9
Kardent	0.0203	0.1225	2.8
Karnico	0.0217	0.1061	3.7
Kartel	0.0203	0.1225	2.8
Katinka	0.0336	0.4200	3.4
Mercator	0.0227	0.3559	5.8
Mercury	0.0203	0.1225	14.2
Nomade	0.0203	0.1225	3.1
Seresta	0.0213	0.2665	4.3
Starga	0.0203	0.1225	2.4
Stefano	0.0203	0.1225	0.6
Valiant	0.0203	0.1225	1.9
Alle	*0.0203*	*0.1225*	*3.34*

6.5.4 Bewaarverliezen door verhoogde temperatuur

Bij bewaartemperaturen die hoger zijn dan 6 °C gaan de aardappelknollen relatief sneller ademen. Bij de ademhaling wordt zetmeel sneller verbruikt waardoor het OWG daalt. Bij hogere temperaturen wordt ook de kiemrust eerder doorbroken waardoor het kiemproces zal beginnen. Zetmeel wordt bij dit kiemproces onder andere als brandstof gebruikt voor de groeiademhaling waardoor er extra bewaarverliezen optreden.

In de module STOCKDAMAGE wordt de temperatuursom bij bewaring per dag bijgehouden. Daarbij wordt een basistemperatuur (TBase) van 6 °C aangehouden, waardoor temperaturen onder de 6 °C niet meetellen in de temperatuursom (Schippers, 1977). Het bewaarverlies (Bewaarverlies$_T$, % d^{-1}) door hogere temperaturen dan 6 °C wordt dan uitgerekend als:

$$Bewaarverlies_T = 0.0001303 \cdot (T - TBase) - 0.00064$$

Met T = gemiddelde dagtemperatuur (°C) en *TBase* = basistemperatuur (6 °C).

6.5.5 Totaal bewaarverlies

Het totale bewaarverlies is de optelling van het onvermijdbare bewaarverlies, het bewaarverlies door rooibeschadiging en het bewaarverlies door verhoogde opslagtemperaturen en kieming:

$$Bewaarverlies_{totaal} = Bewaarverlies_R + Bewaarverlies_{BI} + Bewaarverlies_T$$

6.5.6 Vaste keuzes voor de bewaring van zetmeelaardappelen

Voor het bewaren van zetmeelaardappelen is een aantal vaste technieken gedefinieerd waarmee het model kan rekenen (Tabel 5).

Tabel 5. Bewaartechniek en temperatuurdynamiek.

ID	Bewaartechniek	Temperatuurverloop	Default
0	Geen	Gem. dagtemp.	-
1	Sleuf/Kuil/Silo	Gem. dagtemp. + α °C	α = 2 °C
2	Geventileerde schuur[1]	Gem. nachttemp. + α °C	α = 2 °C
3	Gekoelde schuur	Vaste temperatuur α	α = 6 °C
4	Custom	Opgeven in datafile	-

[1] *niet lager dan 0 °C.*

Referenties

Berge, H.F.M. ten, D.M. Jansen, C. Rappoldt & W. Stol., 1992.
The soil water balance module SAWAH: description and user guide. Simulation Reports CABO-TT 22, Wageningen. 78 pp.

Berge, H.F.M. ten, K. Metselaar, M.J.W. Jansen, E.M de San Agustin & T. Woodhead, 1995.
The SAWAH riceland hydrology model. Water Resources Research 31: 2721-2732.

Dewar, R.C, B.E. Medlyn & R.E. McMurtrtie, 1998.
A mechanistic analysis of light and carbon use efficiencies, Plant Cell and Environment, 21 573-588.

Haren, R.J.F. & D.M. Jansen, 1999.
LINBAL, Light Intereception by Active Leaf Layers. Description and application of a water, nitrogen and late blight limited potato growth model for the Andean Ecoregion. Nota 16, Plant Research International, Wageningen UR, Wageningen. 60 pp + bijlagen.

Haxeltine, A. & I.C. Prentice, 1996.
A general model for the light use efficiency of primary production, Functional Ecology, 10: 551-561.

Ittersum, M.K. van, P.A. Leffelaar, H. van Keulen, M.J. Kropff, L. Bastiaans & J. Goudriaan, 2003.
On approaches and applications of the Wageningen crop models. European Journal of Agronomy 18: 201-234.

Jackson, M.B. & M.C. Drew, 1984.
Effects of flooding on growth and metabolism of herbaceous plants. Pp. 47-128 in T.T. Kozlowski (Ed.), Flooding and plant growth. Academic Press, Londen. 356 pp.

Jansen, D.M., 2002a.
Handleiding voor het simuleren, calibreren, optimaliseren in TIPS-Z. Nota 161, Plant Research International, Wageningen UR, Wageningen. 22 pp. + bijlagen.

Jansen, D.M., 2002b.
Calibratie van model LINBAL voor zetmeelaardappelen. Nota 213, Plant Research International, Wageningen UR, Wageningen. 39 pp.

Jansen, D.M., J.A.R. Davies & J.W. Steenhuizen, 2003a.
Testen van TIPSTAR in de praktijk. Nota 244, Plant Research International, Wageningen UR, Wageningen. 48 pp + bijlagen.

Jansen, D.M., J.A.R. Davies & J.W. Steenhuizen, 2003b.
Gevoeligheid van TIPSTAR voor de waarden van situatie-specifieke invoergegevens. Nota 258, Plant Research International, Wageningen UR, Wageningen. 30 pp. + bijlagen.

Jongschaap, R.E.E., 1996.
Rotask 1.0 – A dynamic simulation model for continous cropping and tillage systems. Reference Manual. Rapport 70, AB-DLO, Wageningen. 40 pp. + bijl.

Keulen, H. van, 1975.
Simulation of water use and herbage growth in arid regions. Simulation Monographs, Pudoc, Wageningen. 176 pp.

Klok, J. & R. Wustman, 2002.
Evaluatie van het teeltbegeleidingssysteem Tipstar. Toepassing van Tipstar in de praktijk. Nota 221, Plant Research International, Wageningen UR, Wageningen. 18 pp. + bijlagen.

Kraalingen, D.W.G. van, 2004.
Pers. Comm.

Kraalingen, D.W.G. van, 1995.
The FSE system for crop simulation, version 2.1. Quantitative Approaches in Systems Analysis No. 1. DLO Research Institute for Agrobiology and Soil Fertility; the C.T. de With graduate school for Production Ecology, Wageningen. 58 pp.

Kraalingen, D.W.G. van & W. Stol., 1997.
Evapotranspiration modules for crop growth simulation. Implementation of the algorithms from Penman, Makkink and Priestley-Taylor. Quantitative Approaches in Systems Analysis No. 11., AB-DLO, Wageningen. 29 pp. + bijl.

Kraalingen, D.W.G. van & C. Rappoldt, 2000.
Reference manual of the Fortran utility library TTUTIL v.4. Report 5, Plant Research International, Wageningen UR, Wageningen. 98 pp.
Penning de Vries, F.W.T., D.M. Jansen, H.F.M. ten Berge & A. Bakema, 1989.
Simulation of ecophysiological processes of growth in several annual crops. Simulation Mongraphs29, Pudoc Wageningen, 271 pp.
Ridder, N. & H. van Keulen, 1995.
Estimating biomass through transfer functions base don simulation model results: a case study for the Sahel. Agricultural Water Management 28: 57-71.
Rodriguez, D., M. van Oijen & A.H.M.C. Schapendonk, 1999.
LINGRA-CC: a sink-source model to simulate the impact of climate change and management on grassland productivity. New Phytologist 144: 359-368.
Spitters C.J.T. & A.H.C.M. Schapendonk, 1990.
Evaluation of breeding strategies for drought tolerance in potato by means of crop growth simulation. Plant and Soil 132(2): 193-203.
Verberne, E.L.J., J. Hasssink, P. de Willigen, J.J.R. Groot & J.A. van Veen, 1990.
Modelling organic matter dynamics in different soils. Netherlands Journal Agricultural Science 38: 221-238.
Verberne, E., G. Dijksterhuis, R.E.E. Jongschaap, H. Hazi, A. Sanou & M. Bonzi, 1995.
Simulation des cultures pluviales au Burkina Faso (CP-BKF3): sorgho, mil et mais. Nota 18, AB-DLO, Wageningen. 53 pp + bijl.
Wit, C.T. de, 1958.
Transpiration and crop yields. Agricultural Research Reports 64.6. Pudoc, Wageningen. 88 p.
Wopereis, M.C.S., B.A.M. Bouma, M.J. Kropff, H.F.M ten Berge & A.R. Maligaya, 1994.
Water use efficiency of flooded rice fields. I. Validation of the soil water balance model SAWAH. Agricultural Water Management 26: 277-289.
Wösten, J.H.M., G.J. Veerman, W.J.M. de Groot & J. Stolte, 2001.
Waterretentie- en doorlatendheidskarakteristieken van boven- en ondergronden in Nederland: de Staringreeks Vernieuwde uitgave 2001. Alterra-rapport 153, Alterra, Wageningen UR, Wageningen, 86 pp.
Wijnholds, K.H., 2002.
Kwaliteit pootgoed Seresta – demo. Verslag in opdracht van AVEBE. PPO, Wageningen UR, Wageningen, 20 pp.
Wijnholds, K.H., 2005.
Demonstratie pootgoedkwaliteit Praktijkmonsters Agrobiokon. PPO Nota 510457. Wageningen UR, Wageningen, 33 pp.

Bijlage I.

Listing modules

I.1. SubroutHine COMBCTL

```
subroutine combctl (ControlTask,Control,ControlFile,LogFileUnit)
implicit none
include '..\GetSoilData\SoilDataSet.inc'
character*(*) ControlTask,Control,ControlFile
integer LogFileUnit
logical Output, First, ICmode, Dowaterstress, donitrostress
logical doblightstress, Dosoilblight, DoPostHarvest, DoEmergence
character*30 SiteName, ManagerName, ParcelName, TreatmentName
character*80 SoilWatModule, CropPosModule, CropReaModule
character*80 EvapModule, WthrModule, WaterUptModule, NDemPosModule
character*80 NdemReaModule, SoilNutModule, NUptModule
Character*80 TotalStressRateModule, TotalStressSTateModule
Character*80 CropDataModule, SoilWatDataModule
character*80 BlightPosModule, BlightReaModule, SoilTempModule
Character*80 ManagementModule, SetSpecParamsModule, DoFinalCalcsModule
character*80 SoilBlightModule, SoilBlightDataFile, PrecInterceptModule
Character*80 PrecInterceptDataFile, PrecInterceptDataFile
character*80 PostHarvestModule, PostHarvestDataFile
character*80 EmergenceModule, EmergenceDataFile
character*80 SunRiSetModule, SunRiSetDataFile
character*80 SetManagementParamsModule
character*80 SoilWatDataFile, PedoTransferDataFile, CropDataFile
Character*80 EvapDataFile, TotalStressDataFile,WaterUptDataFile
Character*80 NDemDataFile, NUptDataFile, SoilNutDataFile
character*80 BlightDataFile, SoilTempDataFile, ManagementDataFile
Character*80 FertilizerDataFile, PesticideDataFile, SoilDataBase
Character*80 SpecParamsIniFile
character*80, parameter :: EmptyString = 'nn'
Character*80, parameter :: SingleBlankString = ' '
character*5 StrTN
integer TN,ICDAT,EDATE
real PPOE
character*80 TFileName,FSESFileName
real ptsstartyear, ptsstartday
integer XNl, IrrDatesMxn,IrrDatesCnt
parameter (IrrDatesMxn=100)
integer    IDATE(IrrDatesMxn)
real       IRVAL(IrrDatesMxn)
integer IrrYear(IrrDatesMxn)
integer IrrDoy(IrrDatesMxn)
integer      StYear,IFlag,ISTN
real         Npl,StDay,FinTim,PrDel,Delt,RainMult,RainAdd
integer      CropStYear,CropStDay, CropHarvestYear, CropHarvestDay,
LaiLeafKillYear, LaiLeafKillDay
integer      H2OStYear,H2OStDay
character*80 WtrDir,CName
character*8  Cntr
integer Unit,ios,tmpi1,I, getun2
logical :: DoMultRuns
save
call getslm (Control, 'domultruns', DoMultRuns, .false.)
if (ControlTask ==  'icinitialize') then
   call getsi (Control,'tn',TN)
   StrTN = ' '
   tmpi1 = 0
   call addint (StrTN,tmpi1,TN)
   call getsi (Control,'EDATE',EDATE)
   call ICDateCnv (EDATE,CropStYear,CropStDay)
   if (CropStYear < 1950) then
      CropStYear = CropStYear + 100
   end if
   Call putsi (Control,'cropstyear',cropstyear)
   Call putsi (Control,'cropstDat',cropstDay)
   Call putsi (Control,'cropstDaY',cropstDay)
   call getsrp (Control,'PPOE',PPOE)
```

```
      Npl = PPOE*10000.
      call getsc (Control,'fsesfilename',FSESFileName)
      Unit = getun2 (20,99,4)
      call icdinit (Unit,FSESFileName,ios)
      if (ios /=  0) call fatalerr ('MAIN',' ')
      call icgksint ('Dowaterstress', Dowaterstress,'TN',StrTN,1)
      call icgksint ('Donitrostress', Donitrostress,'TN',StrTN,1)
      call icgksint ('Doblightstress', Doblightstress,'TN',StrTN,1)
      call icgksint ('Dosoilblight', Dosoilblight,'TN',StrTN,1)
      call icgkscha ('ManagerName', ManagerName, 'TN', StrTN, 1)
      call icgkscha ('SiteName', SiteName, 'TN', StrTN, 1)
      call icgkscha ('ParcelName', ParcelName, 'TN', StrTN, 1)
      call icgkscha ('TreatmentName', TreatmentName, 'TN', StrTN, 1)
      call icgksrea ('startyear',PTSstartyear,'TN',StrTN,1)
      call icgksrea ('startday',PTSstartday,'TN',StrTN,1)
      call icgksrea ('fintim'  ,FinTim     ,'TN',StrTN,1)
      call icgkscha ('Wthr', WthrModule,'TN',StrTN,1)
      call icgkscha ('CropPos', CropPosModule,'TN',StrTN,1)
      call icgkscha ('CropRea', CropReaModule,'TN',StrTN,1)
      call icgkscha ('TotalStressRates', TotalStressRateModule,'TN',StrTN,1)
      call icgkscha('TotalStressStates',TotalStressStateModule,'TN',StrTN,1)
      call icgkscha ('CropData', CropDataModule,'TN',StrTN,1)
      call icgkscha ('CalibratedParams', SpecParamsIniFile, 'TN', StrTN, 1)
      call icgkscha ('Management', ManagementModule,'TN',StrTN,1)
      call icgkscha ('ManagementParameters',Managementdatafile,'TN',StrTN,1)
      call icgkscha ('FertilizerParameters',Fertilizerdatafile,'TN',StrTN,1)
      call icgkscha ('PesticideParameters', Pesticidedatafile,'TN',StrTN,1)
      call icgkscha ('CropParameters', Cropdatafile,'TN',StrTN,1)
      call icgkscha ('PrecIntercept',  PrecInterceptModule,'TN',StrTN,1)
      call LowerC (WthrModule)
      call LowerC (CropPosModule)
      call LowerC (CropReaModule)
      call LowerC (CropDataModule)
      call LowerC (CropDataFile)
      call LowerC (TotalStressRateModule)
      call LowerC (TotalStressStateModule)
      call LowerC (SpecParamsIniFile)
      call LowerC (ManagementModule)
      call LowerC (ManagementDataFile)
      call LowerC (FertilizerDataFile)
      call LowerC (PesticideDataFile)
      call LowerC (PrecInterceptDataFile)
      call putsc (control, 'CropDataFile', CropDataFile)
      if ((DoWaterStress) .or. (DoNitroStress)) then
         call icgkscha ('WaterUpt', WaterUptModule,'TN',StrTN,1)
         call icgkscha ('Evap', EvapModule,'TN',StrTN,1)
         call icgkscha ('SoilWatData', SoilwatDataModule,'TN',StrTN,1)
         call icgkscha ('PedoTransferDataFile',&
                         PedoTransferDataFile,'TN',StrTN,1)
         call icgkscha ('watermodule',SoilWatModule,'TN',StrTN,1)
         call icgkscha ('WaterParameters', SoilWatdatafile,'TN',StrTN,1)
         call icgkscha ('SoilDataBase', SoilDataBase,'TN',StrTN,1)
         call icgkscha ('WaterUptParameters', WaterUptdatafile,'TN',StrTN,1)
         call icgkscha ('EvapParameters', Evapdatafile,'TN',StrTN,1)
         call icgkscha ('TotalstressParameters', &
                         Totalstressdatafile,'TN',StrTN,1)
         call LowerC (WaterUptModule)
         call LowerC (EvapModule)
         call LowerC (SoilWatModule)
         call LowerC (SoilWatDataModule)
         call LowerC (SoilWatDataFile)
         call LowerC (PedoTransferDataFile)
         call LowerC (SoilDataBase)
         call LowerC (WaterUptDataFile)
         call LowerC (EvapDataFile)
         call LowerC (TotalStressDataFile)
         call putsc (control, 'TotalStressDataFile', TotalStressDataFile)
      end if
```

```
      If (DoNitroStress) then
         call icgkscha ('SoilTemp', SoilTempModule,'TN',StrTN,1)
         call icgkscha ('SoilTempParameters', SoilTempdatafile,'TN',StrTN,1)
         call icgkscha ('NdemPos', NdemPosModule,'TN',StrTN,1)
         call icgkscha ('NdemRea', NdemReaModule,'TN',StrTN,1)
         call icgkscha ('SoilNut', SoilNutModule,'TN',StrTN,1)
         call icgkscha ('NUpt', NUptModule,'TN',StrTN,1)
         call icgkscha ('NDemParameters', NDemdatafile,'TN',StrTN,1)
         call icgkscha ('SoilNutParameters', SoilNutdatafile,'TN',StrTN,1)
         call icgkscha ('NuptParameters', Nuptdatafile,'TN',StrTN,1)
         call LowerC (SoilTempModule)
         call LowerC (SoilTempDataFile)
         call LowerC (NdemPosModule)
         call LowerC (NdemReaModule)
         call LowerC (SoilNutModule)
         call LowerC (NUptModule)
         call LowerC (NDemDataFile)
         call LowerC (SoilNutDataFile)
         call LowerC (NuptDataFile)
         call putsc (control, 'NdemDataFile', NDemDataFile)
      end if
      if (DoBlightStress) then
         call icgkscha ('BlightPos', BlightPosModule,'TN',StrTN,1)
         call icgkscha ('BlightRea', BlightReaModule,'TN',StrTN,1)
        call icgkscha ('BlightParameters', Blightdatafile,'TN',StrTN,1)
      end if
      if (Dosoilblight) then
         call icgkscha ('SoilBlight', SoilBlightModule,'TN',StrTN,1)
         call icgkscha ('SoilBlightParameters', &
                         SoilBlightDatafile,'TN',StrTN,1)
         call LowerC (SoilBlightModule)
         call LowerC (SoilBlightDataFile)
      end if
      if (DoPostHarvest) then
         call icgkscha ('PostHarvest', PostHarvestModule,'TN',StrTN,1)
         call icgkscha ('PostHarvestParameters', &
                         PostHarvestDatafile,'TN',StrTN,1)
         call LowerC (PostHarvestModule)
         call LowerC (PostHarvestDataFile)
      end if
      if (DoEmergence) then
         call icgkscha ('Emergence', EmergenceModule,'TN',StrTN,1)
         call icgkscha ('EmergenceParameters', &
                         EmergenceDatafile,'TN',StrTN,1)
         call LowerC (EmergenceModule)
         call LowerC (EmergenceDataFile)
      end if
      call putsc (Control, 'Pesticidedatafile',Pesticidedatafile)
      call putsc (Control, 'Managementdatafile',Managementdatafile)
      call putsc (Control, 'Fertilizerdatafile',Fertilizerdatafile)
      call putsc (Control, 'SoilDataBase', SoilDataBase)
      call putsc (Control, 'ManagerName', ManagerName)
      call putsc (Control, 'SiteName', SiteName)
      call putsc (Control, 'ParcelName', ParcelName)
      call putsc (Control, 'TreatmentName', TreatmentName)
      call putsrp (Control,'StartYear', PTSStartYear)
      call putsc (Control, 'SpecParamsIniFile', SpecParamsIniFile)
      call getsim (Control,'icdat',ICDAT,-99)
      StYear = CropStYear
      StDay = real (CropStDay)
      call icgkscha ('cntr'       ,Cntr       ,'TN',StrTN,1)
      call icgksint ('istn'       ,istn       ,'TN',StrTN,1)
      if (DoWaterStress) then
         if (ICDAT >  -99.) then
            if (ICDAT >  EDATE) then
              call fatalerr (Control,'error')
            end if
            call ICDateCnv (ICDAT,H2OStYear,H2OStDay)
```

```
            if (H2OStYear < 1950) then
               H2OStYear = H2OStYear + 100
            end if
            tmpi1 = 1000*(CropStYear-1900)+CropStDay
            if (tmpi1 >= ICDAT) then
               StYear = H2OStYear
               StDay = real (H2OStDay)
            end if
         else if (ICDAT ==  -99.and.XNl ==  0) then
            H2OStYear = CropStYear
            H2OStDay  = CropStDay
         else
            call fatalerr (Control,'cannot find start condition')
         end if
      end if
      if (Styear > PTSStartyear) Styear = PTSStartyear
      if (StDay > PTSStartday) StDay = PTSStartday
      call icglsrea ('prdel',PrDel,1)
      call icglsrea ('delt' ,Delt ,1)
      call icglsint ('iflag',iflag,1)
      call icglsrea ('RainMult',RainMult,1)
      call icglsrea ('RainAdd',RainAdd,1)
      call getsc (Control,'wtrdir',WtrDir)
      call getsc (Control,'cname' ,CName)
      call getsc (Control,'tfilename',TFileName)
      if (dowaterstress) then
         if ( SoilDataBase == Emptystring) THEN
            if (XNL == 0 .and. soilwatmodule /= Emptystring) then
               call fatalerr (Control,'soil profile not specified: ' // &
               'either modify ptx or modify pts-file')
            end if
            if (XNl >  0) then
               call getsi (Control,'nl',Nl)
               if (Nl >  NlMxn) call fatalerr &
                  (Control,'too many soil layers')
            end if
         else
            Nl = XNl
            call cmdelvar ('Nl')
            call putsi (Control, 'Nl', Nl)
         endif
         call getsi (Control,'xirrdatescnt',IrrDatesCnt)
         if (IrrDatesCnt >  0) then
            call getai  (Control,'idate',IDATE,IrrDatesMxn,i)
            call getarp (Control,'irval',IRVAL,IrrDatesMxn,i)
         end if
         do i=1,IrrDatesCnt
            call ICDateCnv (IDATE(i),IrrYear(i),IrrDoy(i))
         end do
      endif
      call CMDelMod ('runic')
      call putsl (control,'Dowaterstress', Dowaterstress)
      call putsl (control,'Donitrostress', Donitrostress)
      call putsl (control,'Doblightstress', Doblightstress)
      call putsl (control,'Dosoilblight', Dosoilblight)
      call putsl (control,'DoPostHarvest', DoPostHarvest)
      call putsl (control,'DoEmergence'  , DoEmergence)
      call putsi  (Control,'styear',StYear)
      call putsrp (Control,'stDay',StDay)
      call putsrp (Control,'fintim',FinTim)
      call putsrp (Control,'prdel' ,PrDel)
      call putsrp (Control,'delt'  ,Delt)
      call putsc  (Control,'wtrdir',WtrDir)
      call putsc  (Control,'cntr'  ,Cntr)
      call putsi  (Control,'istn'  ,istn)
      call putsi  (Control,'iflag' ,iflag)
      call putsrp (Control,'RainMult',RainMult)
      call putsrp (Control,'RainAdd',RainAdd)
```

```
      call putsrp (Control,'npl',Npl)
      call putsc  (Control,'cname',CName)
      call putsc (Control,'tfilename',TFileName)
      call putsi (Control,'tn',TN)
      if (dowaterstress) then
         if ((SoilWatModule == 'sahe2').or.(SoilWatModule ==  'sahe3')) then
            if ((SoilWatDataFile /= EmptyString) .and. (SoilWatDataFile /= &
                 SingleBlankString)) then
               call cmdelvar( 'WCFC')
               call cmdelvar( 'WCWP')
               call cmdelvar( 'WCAD')
               call cmdelvar( 'WCWU')
               call cmdelvar( 'waterextrpar')
               call loaddatafile (control, SoilWatDataFile, logfileunit)
               call getsim (control, 'swit3', swit3, 0)
               call getarpm (Control,'tkl',tkl,NlMxn,Nl,0.)
               call getarpm (Control, 'frvolumelayer', frvolumelayer, &
                             NlMxn,Nl,0.)
               call getarpm (Control, 'exposedlayer', exposedlayer, &
                             NlMxn,Nl,0.)
            end if
         else if (soilwatmodule == 'wfld') then
            continue
         else if (SoilWatModule ==  'nn') then
            continue
         else if (SoilWatModule == 'sawah') then
           if ((SoilDataBase == EmptyString).or.(SoilDataBase == EmptyString)) then
             call putarp (Control,'kst',kst,Nl)
             call putarp (Control,'vgn',vgn,Nl)
             call putarp (Control,'vga',vga,Nl)
             call putarp (Control,'vgr',vgr,Nl)
             call putarp (Control,'vgl',vgl,Nl)
             call putsrp (Control, 'csc2', csc2)
             call putsrp (Control, 'csa', csa)
             call putsrp (Control, 'csb', csb)
            end if
            if ((SoilWatDataFile/= EmptyString) .and. ((SoilWatDataFile /= &
                 SingleBlankString) .and. (SoilWatDataFile /=EmptyString)))&
               call loaddatafile (control, SoilWatDataFile, logfileunit)
            continue
         else
            call fatalerr (Control,'yet unsupported water balance')
         end if
         call putsi  (Control,'irrdatescnt',IrrDatesCnt)
         if (IrrDatesCnt >  0) then
            call putai  (Control,'irryear',IrrYear,IrrDatesCnt)
            call putai  (Control,'irrdoy' ,IrrDoy ,IrrDatesCnt)
            call putarp (Control,'irramount',IRVAL ,IrrDatesCnt)
         end if
      end if
      call icdterm
      Output = .true.
      First  = .true.
      ICMode = .true.
      call putsl (control,'icmode', icmode)
   else if (ControlTask == 'initialize') then
      call LoadDataFile (Control,ControlFile,LogFileUnit)
      call getslm (Control, 'DoWaterStress', DoWaterStress, .false.)
      call getslm (Control, 'DoNitroStress', DoNitroStress, .false.)
      call getslm (Control, 'DoBlightStress', DoBlightStress, .false.)
      call getslm (Control, 'Dosoilblight'  , Dosoilblight, .false.)
      call getslm (Control, 'DoPostHarvest', DoPostHarvest, .false.)
      call getslm (Control, 'DoEmergence'  , DoEmergence, .false.)
      call getsc  (Control, 'WthrModule'          , WthrModule)
      call getsc  (Control, 'CropPosModule'       , CropPosModule)
      call getsc  (Control, 'CropReaModule'       , CropReaModule)
      call getsc  (Control, 'TotalStressRateModule' , TotalStressRateModule)
      call getsc  (Control, 'TotalStressStateModule',TotalStressStateModule)
```

```
      call getsc  (Control, 'CropDataModule'      , CropDataModule)
      call getsc  (Control, 'ManagementModule'    , ManagementModule)
      call getscm (Control, 'CropDataFile'        , CropDataFile, 'nn')
      call getscm (Control, 'TotalStressDataFile',TotalStressDataFile, 'nn')
      call getscm (control, 'managementdatafile' , managementdatafile, 'nn')
      call getscm (control, 'pesticidedatafile'  , pesticidedatafile, 'nn')
      call getscm (control, 'fertilizerdatafile' , fertilizerdatafile, 'nn')
      if ((DoWaterStress) .or. (DoNitroStress)) then
         call getscm (Control, 'SoilDataBase'      , SoilDataBase, 'nn')
         call getsc  (Control, 'SoilWatModule'     , SoilWatModule)
         call getsc  (Control, 'SoilWatDataModule' , SoilWatDataModule)
         call getscm (Control, 'SoilWatDataFile'   , SoilWatDataFile, 'nn')
         call getscm (Control, 'PedoTransferDataFile', &
                               PedoTransferDataFile, 'nn')
         call getsc  (Control, 'WaterUptModule'      , WaterUptModule)
         call getsc  (Control, 'EvapModule'          , EvapModule)
         call getscm (Control, 'EvapDataFile'        , EvapDataFile, 'nn')
         call getscm (Control, 'WaterUptDataFile'  , WaterUptDataFile, 'nn')
         call LowerC (WaterUptModule)
         call LowerC (EvapModule)
         call LowerC (SoilWatModule)
         call LowerC (SoilWatDataModule)
         call LowerC (SoilWatDataFile)
         call LowerC (PedoTransferDataFile)
         call LowerC (SoilDataBase)
         call LowerC (WaterUptDataFile)
         call LowerC (EvapDataFile)
         call LowerC (TotalStressDataFile)
         if (DoNitroStress) then
            call getsc  (Control, 'NDemPosModule'   , NDemPosModule)
            call getsc  (Control, 'NDemReaModule'   , NDemReaModule)
            call getsc  (Control, 'SoilNutModule'   , SoilNutModule)
            call getsc  (Control, 'NUptModule'      , NuptModule)
            call getscm (Control, 'NDemDataFile'    , NDemDataFile, 'nn')
            call getscm (Control, 'SoilNutDataFile' , SoilNutDataFile, 'nn')
            call getscm (Control, 'NUptDataFile'    , NuptDataFile, 'nn')
            call getsc  (Control, 'SoilTempModule'  , SoilTempModule)
            call getscm (Control, 'SoilTempDataFile', SoilTempDataFile, nn')
         endif
      endif
      if (DoBlightStress) then
         call getsc  (Control, 'BlightPosModule', BlightPosModule)
         call getsc  (Control, 'BlightReaModule', BlightReaModule)
         call getscm (Control, 'BlightDataFile'      , BlightDataFile, 'nn')
      endif
      if (Dosoilblight) then
         call getsc  (Control, 'SoilBlightModule'  ,SoilBlightModule)
         call getscm (Control, 'SoilBlightDataFile',SoilBlightDataFile,'nn')
         call LowerC (SoilBlightModule)
         call LowerC (SoilBlightDataFile)
      endif
      if (DoPostHarvest) then
         call getsc  (Control, 'PostHarvestModule'  ,PostHarvestModule)
         call getscm (Control, 'PostHarvestDataFile',PostHarvestDataFile, &
                      'nn')
         call LowerC (PostHarvestModule)
         call LowerC (PostHarvestDataFile)
      endif
      if (DoEmergence) then
         call getsc  (Control, 'EmergenceModule'   , EmergenceModule)
         call getscm (Control, 'EmergenceDataFile' , EmergenceDataFile,'nn')
         call LowerC (EmergenceModule)
         call LowerC (EmergenceDataFile)
      endif
      Output = .true.
      First  = .true.
      ICMode = .false.
      call putsl (control,'icmode', icmode)
```

```
406   else if (ControlTask == 'dynamic') then
407      if (First) then
408         call getscm (Control, 'SetSpecParamsModule', SetSpecParamsModule, &
409                      'nn')
410         call getscm (Control, 'DoFinalCalcsModule'  , DoFinalCalcsModule, &
411                      'nn')
412         call getscm (Control, 'SetManagementParamsModule', &
413                       SetManagementParamsModule, 'nn')
414         if (( ManagementModule /= EmptyString) .and. (ManagementModule /= &
415                       SingleBlankString)) &
416            call StartModule (ManagementModule, ' ', Output)
417         if (( CropDataModule == EmptyString) .or. (CropDataModule == &
418               SingleBlankString) .and. ( CropDataFile   /= EmptyString) &
419               .and. (CropDataFile   /= SingleBlankString)) &
420               call loaddatafile (control, CropDataFile, logfileunit)
421         if ((DoWaterStress) .or. (DoNitrostress)) then
422            if (( CropDataModule == EmptyString) .or. (CropDataModule == &
423                  SingleBlankString) .and. ( TotalStressDataFile /=  &
424                  EmptyString) .and. (TotalStressDataFile /= &
425                  SingleBlankString)) &
426               call loaddatafile (control, TotalStressDataFile, logfileunit)
427            if (( SoilWatDataFile /= EmptyString) .and. (SoilWatDataFile /=&
428                 SingleBlankString)) then
429               call loaddatafile (control, SoilWatDataFile, logfileunit)
430               call getsim(control, 'swit3', swit3, 0)
431            end if
432            if (( PedoTransferDataFile /= EmptyString) .and. &
433                 (PedoTransferDataFile /= SingleBlankString)) then
434               call loaddatafile (control, PedoTransferDataFile,logfileunit)
435            end if
436            if (( EvapDataFile /= EmptyString) .and. (EvapDataFile /= &
437               SingleBlankString)) &
438               call loaddatafile (control, EvapDataFile, logfileunit)
439            if (( WaterUptDataFile /= EmptyString) .and. (WaterUptDataFile &
440               /= SingleBlankString)) &
441               call loaddatafile (control, WaterUptDataFile, logfileunit)
442            if (DoNitroStress) then
443               if ((SoilTempDataFile /= EmptyString).and.(SoilTempDataFile &
444                  /= SingleBlankString)) &
445                  call loaddatafile (control, SoilTempDataFile, logfileunit)
446               if ((CropDataModule == EmptyString) .or. (CropDataModule == &
447                  SingleBlankString) .and. ( NDemDataFile   /= EmptyString)&
448                  .and. (NDemDataFile   /= SingleBlankString)) &
449                  call loaddatafile (control, NDemDataFile, logfileunit)
450               if (( NuptDataFile /= EmptyString) .and. (NuptDataFile /= &
451                    SingleBlankString)) &
452                  call loaddatafile (control, NuptDataFile, logfileunit)
453               if ((SoilNutDataFile /= EmptyString) .and. (SoilNutDataFile &
454                  /= SingleBlankString)) &
455                  call loaddatafile (control, SoilNutDataFile, logfileunit)
456            end if
457         end if
458         if (DoBlightStress) then
459            if (( BlightDataFile /= EmptyString) .and. (BlightDataFile /= &
460               SingleBlankString)) &
461               call loaddatafile (control, BlightDataFile, logfileunit)
462         end if
463         if (Dosoilblight) then
464            if ((SoilBlightDataFile /= EmptyString) .and. &
465                 (SoilBlightDataFile /= SingleBlankString)) &
466               call loaddatafile (control, SoilBlightDataFile, logfileunit)
467         end if
468         if (DoPostHarvest) then
469            if ((PostHarvestDataFile /= EmptyString) .and. &
470              (PostHarvestDataFile /= SingleBlankString)) &
471               call loaddatafile (control, PostHarvestDataFile, logfileunit)
472         end if
473         if (DoEmergence) then
```

```
474            if ((EmergenceDataFile /= EmptyString).and.(EmergenceDataFile &
475               /= SingleBlankString)) &
476               call loaddatafile (control, EmergenceDataFile, logfileunit)
477         end if
478         if (ICMode) then
479         else
480            call LoadDataFile (Control,ControlFile,LogFileUnit)
481         endif
482         call cmdelvar(SoilwatDataFile)
483         call cmdelvar(PedoTransferDataFile)
484         call cmdelvar(EvapDataFile)
485         call cmdelvar(WaterUptDataFile)
486         call cmdelvar(SoilTempDataFile)
487         call cmdelvar(NuptDataFile)
488         call cmdelvar(SoilNutDataFile)
489         call cmdelvar(BlightDataFile)
490         if (( CropDataModule /= EmptyString) .and. (CropDataModule /= &
491               SingleBlankString)) &
492            call StartModule (CropDataModule, ' ', Output)
493         if ((DoWaterStress) .or. (DoNitroStress)) then
494            if ((SoilDataBase /= Emptystring) .and. (SoilDataBase /= &
495               SingleBLankString)) then
496               call StartModule (SoilWatDataModule, ' ', Output)
497            else if (ICMode == .false. ) then
498               if (( SoilWatDataModule /= EmptyString) .and. &
499                  (SoilWatDataModule /= SingleBlankString)) &
500                  call StartModule (SoilWatDataModule, ' ', Output)
501            end if
502         end if
503         if (ICMODE) then
504            call StartModule ('tdata',' ',.true.)
505         end if
506         if (( SetSpecParamsModule /= EmptyString) .and. &
507            (SetSpecParamsModule /= SingleBlankString)) &
508            call StartModule (SetSpecParamsModule, ' ', Output)
509         if (( WthrModule /= EmptyString) .and. (WthrModule /= &
510            SingleBlankString)) &
511            Call StartModule (WthrModule,' ' , Output)
512         if ((DoWaterStress).or.(DoNitroStress)) then
513            if ( EvapModule .eq. 'etfad') &
514               Call StartModule ('asfao',' ' , Output)
515            if (( EvapModule /= EmptyString) .and. (EvapModule /= &
516               SingleBlankString)) &
517               Call StartModule (EvapModule,' ' , Output)
518         end if
519         if (( TotalStressStateModule /= EmptyString) .and. &
520              (TotalStressStateModule /= SingleBlankString)) &
521            call StartModule (TotalStressStateModule,    ' ' , Output)
522         if (( CropPosModule /= EmptyString) .and. (CropPosModule /= &
523            SingleBlankString)) &
524            Call StartModule (CropPosModule, ' ', Output)
525         if (ICMODE) then
526            if (IrrDatesCnt >  0) then
527               call StartModule ('sirri', ' ', Output)
528            end if
529         end if
530         if (( SetManagementParamsModule /= EmptyString) .and. &
531            (SetManagementParamsModule /= SingleBlankString)) &
532            Call StartModule (SetManagementParamsModule, ' ', Output)
533         if ((DoWaterStress).or.(DoNitroStress)) then
534            if (( WaterUptModule /= EmptyString) .and. (WaterUptModule /= &
535               SingleBlankString)) &
536               Call StartModule (WaterUptModule, ' ', Output)
537            if (( SoilWatModule /= EmptyString) .and. (SoilWatModule /= &
538               SingleBlankString)) &
539               call StartModule (SoilWatModule, ' ' ,Output)
540         end if
541         if (DoNitroStress) then
```

```
542            if (( SoilTempModule /= EmptyString) .and. (SoilTempModule /= &
543               SingleBlankString)) &
544               call StartModule (SoilTempModule, ' ' , Output)
545            if (( NdemPosModule /= EmptyString) .and. (NdemPosModule /= &
546               SingleBlankString)) &
547               call StartModule (NdemPosModule,    ' ' , Output)
548            if (( NuptModule /= EmptyString) .and. (NuptModule /= &
549               SingleBlankString)) &
550               call StartModule (NUptModule,    ' ' , Output)
551            if (( SoilNutModule /= EmptyString) .and. (SoilNutModule /= &
552               SingleBlankString)) &
553               call StartModule (SoilNutModule, ' ' , Output)
554         end if
555         if (DoBlightStress) then
556            if (( BlightPosModule /= EmptyString) .and.(BlightPosModule /= &
557               SingleBlankString)) &
558               call StartModule (BlightPosModule, ' ' , Output)
559         end if
560         if (Dosoilblight) then
561            if (( PrecInterceptModule /= EmptyString) .and. &
562               (PrecInterceptModule /= SingleBlankString)) &
563               call StartModule (PrecInterceptModule, ' ' , Output)
564            if (( SoilBlightModule /= EmptyString).and.(SoilBlightModule/= &
565               SingleBlankString)) &
566               call StartModule (SoilBlightModule, ' ' , Output)
567         end if
568         if (DoEmergence) then
569            if (( EmergenceModule /= EmptyString) .and. (EmergenceModule/= &
570               SingleBlankString)) &
571               call StartModule (EmergenceModule, ' ' , Output)
572         end if
573         if (( TotalStressRateModule /= EmptyString) .and. &
574            (TotalStressRateModule /= SingleBlankString)) &
575            call StartModule (TotalStressRateModule,    ' ' , Output)
576         if (( CropReaModule /= EmptyString) .and. (CropReaModule /= &
577            SingleBlankString)) &
578            call StartModule (CropReaModule, ' ' , Output)
579         if (DoPostHarvest) then
580            if (( PostHarvestModule /= EmptyString) .and. &
581               (PostHarvestModule/= SingleBlankString)) &
582               call StartModule (PostHarvestModule, ' ' , Output)
583         end if
584         if (DoNitroStress) then
585            if (( NdemReaModule /= EmptyString) .and. (NdemReaModule /= &
586               SingleBlankString)) &
587               call StartModule (NdemReaModule,' ', Output)
588         end if
589         if (DoBlightStress) then
590            if ((BlightReaModule /= EmptyString) .and. (BlightReaModule /= &
591               SingleBlankString)) &
592               call StartModule (BlightReaModule, ' ' , Output)
593         end if
594         if ((DoFinalCalcsModule /= EmptyString) .and. (DoFinalCalcsModule &
595            /= SingleBlankString)) &
596            Call StartModule (DoFinalCalcsModule, ' ', Output)
597         if (DoMultRuns) call StartModule ('getrerunsresvar', ' ' , Output)
598         First = .false.
599      end if
600   else if (ControlTask == 'stopmodules?') then
601      continue
602   else
603      call fatalerr (Control,'wrong task')
604   end if
605   return
606   end subroutine combctl
```

I.2. Subroutine EMERGENCE

```
subroutine emergence(xNewTask,xModule,xLogFileUnit)
implicit none
include '..\GetCropData\CropDataSet.inc'
character*(*) xNewTask,xModule
integer       xLogFileUnit
real ::       Delt
real ::       Doy
real :: AlphaKiem, MaxKiem, TempSumKiem, TempSumKiem80
real :: Alp1, Alp2, Alp3, Alp4, Alp5, Alp6, Alp7 ,Alp8 ,Alp9 ,Alp10, &
        Alp11,Alp12,Alp13,Alp14,Alp15,Alp16,Alp17,Alp18,Alp19,Alp20, &
        Alp21,Alp22,Alp23,Alp24,Alp25,Alp26,Alp27,Alp28,Alp29,Alp30
real :: MxK1, MxK2, MxK3, MxK4, MxK5, MxK6, MxK7, MxK8, MxK9, MxK10, &
        MxK11,MxK12,MxK13,MxK14,MxK15,MxK16,MxK17,MxK18,MxK19,MxK20, &
        MxK21,MxK22,MxK23,MxK24,MxK25,MxK26,MxK27,MxK28,MxK29,MxK30
real :: TmS1, TmS2, TmS3, TmS4, TmS5, TmS6, TmS7, TmS8, TmS9, TmS10, &
        TmS11,TmS12,TmS13,TmS14,TmS15,TmS16,TmS17,TmS18,TmS19,TmS20, &
        TmS21,TmS22,TmS23,TmS24,TmS25,TmS26,TmS27,TmS28,TmS29,TmS30
real :: Beschadiging, Zilverschurft, Fusarium, Rot, Rhizoctonia, &
        Schurft, Waardering, Veroudering, Kiemlengte, OWG
real :: TempSumx, tmda
integer :: IsAfkiem, IsTempCont_NT, IsTempCont_MV, IsTempCont_DW, IsTempCont_RV,
IsTempCont_MK, &
           IsBewaar_C, IsBewaar_G, IsBewaar_H, IsBewaar_K, &
           IsDrogen_NT, IsDrogen_TR, IsDrogen_TV
integer :: CropPlantingDay, CropPlantingYear, idoy, iyear, DateCmp
logical :: emerged
character(len=13) :: OldTask, EmptyString
integer :: DTFSECMP
save
if (OldTask == EmptyString) OldTask = 'terminate'
call chktsk3 (xNewTask,OldTask,xModule)
OldTask = xNewTask
if (xNewTask == 'initialize') then
   call Logmessage (xLogFileUnit,9,xModule,'0.0')
   call getsl (xModule, 'DoEmergence', DoEmergence)
   Emerged = .false.
   If (DoEmergence) then
      Call getsrp (xModule,'delt',Delt)
      Call getsrp (xModule, 'alphakiem1'     , Alp1)
      Call getsrp (xModule, 'alphakiem2'     , Alp2)
      Call getsrp (xModule, 'alphakiem3'     , Alp3)
      Call getsrp (xModule, 'alphakiem4'     , Alp4)
      Call getsrp (xModule, 'alphakiem5'     , Alp5)
      Call getsrp (xModule, 'alphakiem6'     , Alp6)
      Call getsrp (xModule, 'alphakiem7'     , Alp7)
      Call getsrp (xModule, 'alphakiem8'     , Alp8)
      Call getsrp (xModule, 'alphakiem9'     , Alp9)
      Call getsrp (xModule, 'alphakiem10'    , Alp10)
      Call getsrp (xModule, 'alphakiem11'    , Alp11)
      Call getsrp (xModule, 'alphakiem12'    , Alp12)
      Call getsrp (xModule, 'alphakiem13'    , Alp13)
      Call getsrp (xModule, 'alphakiem14'    , Alp14)
      Call getsrp (xModule, 'alphakiem15'    , Alp15)
      Call getsrp (xModule, 'alphakiem16'    , Alp16)
      Call getsrp (xModule, 'alphakiem17'    , Alp17)
      Call getsrp (xModule, 'alphakiem18'    , Alp18)
      Call getsrp (xModule, 'alphakiem19'    , Alp19)
      Call getsrp (xModule, 'alphakiem20'    , Alp20)
      Call getsrp (xModule, 'alphakiem21'    , Alp21)
      Call getsrp (xModule, 'alphakiem22'    , Alp22)
      Call getsrp (xModule, 'alphakiem23'    , Alp23)
      Call getsrp (xModule, 'alphakiem24'    , Alp24)
      Call getsrp (xModule, 'alphakiem25'    , Alp25)
      Call getsrp (xModule, 'alphakiem26'    , Alp26)
      Call getsrp (xModule, 'alphakiem27'    , Alp27)
```

```
      Call getsrp (xModule, 'alphakiem28'    , Alp28)
      Call getsrp (xModule, 'maxkiem1'     , MxK1)
      Call getsrp (xModule, 'maxkiem2'     , MxK2)
      Call getsrp (xModule, 'maxkiem3'     , MxK3)
      Call getsrp (xModule, 'maxkiem4'     , MxK4)
      Call getsrp (xModule, 'maxkiem5'     , MxK5)
      Call getsrp (xModule, 'maxkiem6'     , MxK6)
      Call getsrp (xModule, 'maxkiem7'     , MxK7)
      Call getsrp (xModule, 'maxkiem8'     , MxK8)
      Call getsrp (xModule, 'maxkiem9'     , MxK9)
      Call getsrp (xModule, 'maxkiem10'    , MxK10)
      Call getsrp (xModule, 'maxkiem11'    , MxK11)
      Call getsrp (xModule, 'maxkiem12'    , MxK12)
      Call getsrp (xModule, 'maxkiem13'    , MxK13)
      Call getsrp (xModule, 'maxkiem14'    , MxK14)
      Call getsrp (xModule, 'maxkiem15'    , MxK15)
      Call getsrp (xModule, 'maxkiem16'    , MxK16)
      Call getsrp (xModule, 'maxkiem17'    , MxK17)
      Call getsrp (xModule, 'maxkiem18'    , MxK18)
      Call getsrp (xModule, 'maxkiem19'    , MxK19)
      Call getsrp (xModule, 'maxkiem20'    , MxK20)
      Call getsrp (xModule, 'maxkiem21'    , MxK21)
      Call getsrp (xModule, 'maxkiem22'    , MxK22)
      Call getsrp (xModule, 'maxkiem23'    , MxK23)
      Call getsrp (xModule, 'maxkiem24'    , MxK24)
      Call getsrp (xModule, 'maxkiem25'    , MxK25)
      Call getsrp (xModule, 'maxkiem26'    , MxK26)
      Call getsrp (xModule, 'maxkiem27'    , MxK27)
      Call getsrp (xModule, 'maxkiem28'    , MxK28)
      Call getsrp (xModule, 'TempSumKiem1'     , TmS1)
      Call getsrp (xModule, 'TempSumKiem2'     , TmS2)
      Call getsrp (xModule, 'TempSumKiem3'     , TmS3)
      Call getsrp (xModule, 'TempSumKiem4'     , TmS4)
      Call getsrp (xModule, 'TempSumKiem5'     , TmS5)
      Call getsrp (xModule, 'TempSumKiem6'     , TmS6)
      Call getsrp (xModule, 'TempSumKiem7'     , TmS7)
      Call getsrp (xModule, 'TempSumKiem8'     , TmS8)
      Call getsrp (xModule, 'TempSumKiem9'     , TmS9)
      Call getsrp (xModule, 'TempSumKiem10'    , TmS10)
      Call getsrp (xModule, 'TempSumKiem11'    , TmS11)
      Call getsrp (xModule, 'TempSumKiem12'    , TmS12)
      Call getsrp (xModule, 'TempSumKiem13'    , TmS13)
      Call getsrp (xModule, 'TempSumKiem14'    , TmS14)
      Call getsrp (xModule, 'TempSumKiem15'    , TmS15)
      Call getsrp (xModule, 'TempSumKiem16'    , TmS16)
      Call getsrp (xModule, 'TempSumKiem17'    , TmS17)
      Call getsrp (xModule, 'TempSumKiem18'    , TmS18)
      Call getsrp (xModule, 'TempSumKiem19'    , TmS19)
      Call getsrp (xModule, 'TempSumKiem20'    , TmS20)
      Call getsrp (xModule, 'TempSumKiem21'    , TmS21)
      Call getsrp (xModule, 'TempSumKiem22'    , TmS22)
      Call getsrp (xModule, 'TempSumKiem23'    , TmS23)
      Call getsrp (xModule, 'TempSumKiem24'    , TmS24)
      Call getsrp (xModule, 'TempSumKiem25'    , TmS25)
      Call getsrp (xModule, 'TempSumKiem26'    , TmS26)
      Call getsrp (xModule, 'TempSumKiem27'    , TmS27)
      Call getsrp (xModule, 'TempSumKiem28'    , TmS28)
      Call getsrp (xModule, 'Beschadiging'   , Beschadiging)
      Call getsrp (xModule, 'Zilverschurft'  , Zilverschurft)
      Call getsrp (xModule, 'Fusarium'       , Fusarium)
      Call getsrp (xModule, 'Rot'            , Rot)
      Call getsrp (xModule, 'Rhizoctonia'    , Rhizoctonia)
      Call getsrp (xModule, 'Schurft'        , Schurft)
      Call getsrp (xModule, 'Waardering'     , Waardering)
      Call getsrp (xModule, 'Veroudering'    , Veroudering)
      Call getsrp (xModule, 'Kiemlengte'     , Kiemlengte)
      Call getsrp (xModule, 'OWG'            , OWG)
      Call getsi  (xModule, 'IsAfkiem',      IsAfkiem)
```

```
      Call getsi  (xModule, 'IsTempCont_NT', IsTempCont_NT)
      Call getsi  (xModule, 'IsTempCont_MV', IsTempCont_MV)
      Call getsi  (xModule, 'IsTempCont_DW', IsTempCont_DW)
      Call getsi  (xModule, 'IsTempCont_RV', IsTempCont_RV)
      Call getsi  (xModule, 'IsTempCont_MK', IsTempCont_MK)
      Call getsi  (xModule, 'IsBewaar_C',    IsBewaar_C)
      Call getsi  (xModule, 'IsBewaar_G',    IsBewaar_G)
      Call getsi  (xModule, 'IsBewaar_H',    IsBewaar_H)
      Call getsi  (xModule, 'IsBewaar_K',    IsBewaar_K)
      Call getsi  (xModule, 'IsDrogen_NT',   IsDrogen_NT)
      Call getsi  (xModule, 'IsDrogen_TR',   IsDrogen_TR)
      Call getsi  (xModule, 'IsDrogen_TV',   IsDrogen_TV)
      call EmerPars(Alp1, Alp2, Alp3, Alp4, Alp5, Alp6, Alp7, Alp8,  &
                    Alp9, Alp10,Alp11,Alp12,Alp13,Alp14,Alp15,Alp16, &
                    Alp17,Alp18,Alp19,Alp20,Alp21,Alp22,Alp23,Alp24, &
                    Alp25,Alp26,Alp27,Alp28,Alp29,Alp30,IsAfkiem,    &
                    IsTempCont_NT, IsTempCont_MV, IsTempCont_DW,     &
                    IsTempCont_RV, IsTempCont_MK, IsBewaar_C,        &
                    IsBewaar_G, IsBewaar_H, IsBewaar_K, IsDrogen_NT, &
                    IsDrogen_TR, IsDrogen_TV, Beschadiging,          &
                    Zilverschurft, Fusarium, Rot, Rhizoctonia, Schurft, &
                    Waardering, Veroudering, Kiemlengte, OWG, AlphaKiem )
      call EmerPars(MxK1, MxK2, MxK3, MxK4, MxK5, MxK6, MxK7, MxK8,  &
                    MxK9, MxK10,MxK11,MxK12,MxK13,MxK14,MxK15,MxK16, &
                    MxK17,MxK18,MxK19,MxK20,MxK21,MxK22,MxK23,MxK24, &
                    MxK25,MxK26,MxK27,MxK28,MxK29,MxK30, IsAfkiem,   &
                    IsTempCont_NT, IsTempCont_MV, IsTempCont_DW,     &
                    IsTempCont_RV, IsTempCont_MK, IsBewaar_C,        &
                    IsBewaar_G, IsBewaar_H, IsBewaar_K, IsDrogen_NT, &
                    IsDrogen_TR, IsDrogen_TV, Beschadiging,          &
                    Zilverschurft, Fusarium, Rot, Rhizoctonia, Schurft, &
                    Waardering, Veroudering, Kiemlengte, OWG, MaxKiem )
      call EmerPars(TmS1, TmS2, TmS3, TmS4, TmS5, TmS6, TmS7, TmS8,  &
                    TmS9, TmS10,TmS11,TmS12,TmS13,TmS14,TmS15,TmS16, &
                    TmS17,TmS18,TmS19,TmS20,TmS21,TmS22,TmS23,TmS24, &
                    TmS25,TmS26,TmS27,TmS28,TmS29,TmS30, IsAfkiem,   &
                    IsTempCont_NT, IsTempCont_MV, IsTempCont_DW,     &
                    IsTempCont_RV, IsTempCont_MK, IsBewaar_C,        &
                    IsBewaar_G, IsBewaar_H, IsBewaar_K, IsDrogen_NT, &
                    IsDrogen_TR, IsDrogen_TV, Beschadiging,          &
                    Zilverschurft, Fusarium, Rot, Rhizoctonia, Schurft, &
                    Waardering, Veroudering, Kiemlengte, OWG,TempSumKiem)
      TempSumKiem80 = TempSumKiem - alog(0.2) / AlphaKiem
      MaxKiem = min (1., MaxKiem)
      TempSumX = 0.
      call getsi (xModule, 'CropPlantingDay',  CropPlantingDay)
      call getsi (xModule, 'CropPlantingYear', CropPlantingYear)
      call getsi (xModule, 'CropStDay',  CropStDay)
      call getsi (xModule, 'CropStYear', CropStYear)
      call putsi ('master', 'CropStDay', 9999)
   else
      return
   end if
else if (xNewTask == 'do_rates') then
   If (DoEmergence) call getsrt (xModule,'tmda' , tmda)
else if (xNewTask == 'do_states') then
   If (DoEmergence) then
      Call getsrt (xModule, 'doy', doy)
      call getsi (xModule,'idoy',idoy)
      call getsi (xModule,'iyear',iyear)
      DateCmp  = dtfsecmp (CropPlantingYear,CropPlantingDay,iyear,idoy-1)
      if ((DateCmp >= 0) .and. (.not.(Emerged))) then
         TempSumX = TempSumX + max(0., (TmDa - 5.))
         If ((TempSumX >= TempSumKiem80)) then
            call putsi ('master', 'CropStDay', idoy)
            call putsi ('master', 'CropStYear', iyear)
            call getsrp (xModule, 'npl', npl)
            npl = MaxKiem * NPL
```

```
202                call putsrp ('master', 'npl', npl)
203                Emerged = .true.
204             end if
205          Elseif(.not.(Emerged)) then
206             call putsi ('master', 'CropStDay', 9999)
207          Endif
208       else
209          return
210       end if
211    end if
212    Return
213    End subroutine Emergence
```

I.3. Subroutine EMERPARS

```
subroutine EmerPars(Par1, Par2, Par3, Par4, Par5, Par6, Par7, Par8,  &
                    Par9, Par10,Par11,Par12,Par13,Par14,Par15,Par16, &
                    Par17,Par18,Par19,Par20,Par21,Par22,Par23,Par24, &
                    Par25,Par26,Par27,Par28,Par29,Par30, IsAfkiem,   &
                    IsTempCont_NT, IsTempCont_MV, IsTempCont_DW,     &
                    IsTempCont_RV, IsTempCont_MK, IsBewaar_C,        &
                    IsBewaar_G, IsBewaar_H, IsBewaar_K, IsDrogen_NT, &
                    IsDrogen_TR, IsDrogen_TV, Beschadiging,          &
                    Zilverschurft, Fusarium, Rot, Rhizoctonia, Schurft, &
                    Waardering, Veroudering, Kiemlengte, OWG, X )
implicit none
integer :: IsAfkiem, IsTempCont_NT, IsTempCont_MV, IsTempCont_DW, IsTempCont_RV,
IsTempCont_MK
integer :: IsBewaar_C, IsBewaar_G, IsBewaar_H, IsBewaar_K
integer :: IsDrogen_NT, IsDrogen_TR, IsDrogen_TV
real :: X
real :: Par1, Par2, Par3, Par4, Par5, Par6, Par7, Par8, Par9, Par10
real :: Par11,Par12,Par13,Par14,Par15,Par16,Par17,Par18,Par19,Par20
real :: Par21,Par22,Par23,Par24,Par25,Par26,Par27,Par28,Par29,Par30
real :: eff_Afkiemen_TempCont, eff_Afkiemen_Bewaar, eff_Afkiemen_Drogen
real :: eff_Afkiemen_Ziekten, eff_Afkiemen_Conditie, eff_Conditie
real :: eff_Temperatuur_Controle, eff_Bewaar, eff_Drogen, eff_Ziekten
real :: Beschadiging, Zilverschurft, Fusarium, Rot, Rhizoctonia, Schurft
real :: Waardering, Veroudering, Kiemlengte, OWG
eff_Afkiemen_TempCont  = Par2 * IsAfkiem
eff_Afkiemen_Bewaar    = Par3 * IsAfkiem
eff_Afkiemen_Drogen    = Par4 * IsAfkiem
eff_Afkiemen_Ziekten   = Par5 * IsAfkiem
eff_Afkiemen_Conditie  = Par6 * IsAfkiem
eff_Temperatuur_Controle = (Par7 * IsTempCont_NT + Par8 * IsTempCont_MV &
         + Par9 * IsTempCont_DW + Par10 * IsTempCont_RV + Par11 * &
         IsTempCont_MK )
eff_Bewaar = (Par12 * IsBewaar_C + Par13 * IsBewaar_G + Par14 * &
         IsBewaar_H + Par15 * IsBewaar_K )
eff_Drogen = ( Par16  * IsDrogen_NT + Par17 * IsDrogen_TR + Par18 * &
         IsDrogen_TV )
eff_Ziekten = (Par19 * Beschadiging + Par20 * Fusarium + Par21 * Rot + &
         Par22 * Rhizoctonia  + Par23 * Schurft + Par24 * Zilverschurft)
eff_Conditie = (Par25 * Waardering + Par26 * Veroudering + Par27 * &
                Kiemlengte + Par28 * OWG )
X = Par1 * (1 + (1 + eff_Afkiemen_TempCont) * eff_Temperatuur_Controle)*&
           (1 + (1 + eff_Afkiemen_Bewaar)   * eff_Bewaar)   *           &
           (1 + (1 + eff_Afkiemen_Drogen)   * eff_Drogen)   *           &
           (1 + (1 + eff_Afkiemen_Ziekten)  * eff_Ziekten) *            &
           (1 + (1 + eff_Afkiemen_Conditie) * eff_Conditie)
end subroutine EmerPars
```

I.4. CROPDATASET.INC

```
CROPDATASET.INC
integer :: cropstyear, cropstday, cropharvestday, cropharvestyear,
integer :: laileafkillday, laileafkillyear, LeafClCnt, LeafClCntI
integer :: TMEffGrowthTBL, CNamesCnt
integer, parameter :: CNamesMxn = 25
integer, parameter :: LeafClMxn = 300
integer, parameter :: TMTableMxn = 100
real :: TMSumI, LeafArPlI, LeafDeadWtI, StemWtI, TuberWtI, RootWtI
real :: RootDepthI, RootHeightI, nstabconci, nstructconci, nsolconci
real :: LeafArPl, LAILive, LAIDead, LaiTot, LaiEff, TMSum, LeafWt
real :: LafDeadWt, StemWt, StemDeadWt, otWt, RootDepth, RootHeight
real :: TuberWt, RootDeadWt, TuberDeadWt, TuberFreshWt, LeafAr
real :: LeafDeadAr, NPL, TMEffGrowthTB(TMTableMxn)
real :: SLA, RGRL, ECPDF, ECPDFDead, ECPDFStem, relLIntleafdead
real :: relLIntstemdead, ECPDFRed, ECPDFGrn, EffPhotStem
real :: PARFraction, TMBase, LUE, LeafPar, LeafStemRatio
real :: TuberFreshDryRatio, RootDepthCropMx, RootDepthGrowthPar
real :: KstrsLUE, KstrsRGRL, KstrsAGE, KstrsAr, BetastrsAr, Bage, Gage
real :: KageLUE, BageLUE, ksynstab, kdestab,fractubstruc,fracabovestruc
real :: RelLossDeadLeaves, RelLossDeadStems,RelLossDeadTubers
real :: RelLossDeadRoots, alphaT, betaT, gammaT, deltaT, alphaLvAgeT
real :: betaLvAgeT, MaxNupRateMax, Lfltar, knlue, knrgrl, noffset
real :: alphaspad,betaspad, gammaspad, deltaspad, nspad, mspad, maxspad
real :: TMSumLeafGrowth, LeafAgeMx, LeafArGrowthRef, WaterExtrPar
real :: RainIntPar
real, dimension(LeafClMxn) :: leafagecl, rleafageclp, RLeafAgeCl, &
      stressindexlvagcl, nstabclassconc, nstrucclassconc, nstablivecl,&
      nstruclivecl, nsollivecl, distrdmleafcl, LeafArDeadCl, &
      LeafWtDeadCl, FreeDMLeafCl, stressindexlvagclP, nstablosscl, &
      nstruclosscl, nstabdeadcl, nstrucdeadcl, LeafArLiveCl, &
      LeafWtLiveCl, stressindexlvdyingcl, RDyingLeafCl, LeafWtLossCl, &
      LeafArLossCl, FrSenescence, BlightStressIndexLvAgCl, &
      NitroStressIndexLvDyingCl, BlightStressIndexLvDyingCl, &
      RLeafWtDeadClp, RLeafWtLiveClp, RLeafArDeadClp, RLeafArLiveClp
character(LEN=80) :: CNames(CNamesMxn), CNames2(CNamesMxn),CName
logical :: DoPostHarvest, DoEmergence, LeafAliveCl(LeafClMxn)
common /CropDataSet/ cropstyear, cropstday, cropharvestyear, &
      cropharvestday, laileafkillyear, laileafkillday, LeafArPl, &
      LAILive, LaiEff,LAIDead, LAITot, TMSum, LeafWt, LeafDeadWt, &
      StemWt, RootWt, RootDepth, RootHeight, TuberWt, TuberFreshWt, &
      LeafAr, LeafClCnt
common /CropTemp/leafagecl, rleafageclp,RLeafAgeCl, stressindexlvagcl,&
      stressindexlvagclP, LeafArLiveCl, LeafWtLiveCl, nstabclassconc, &
      nstrucclassconc,nstablivecl, nstruclivecl, nsollivecl, &
      nstabdeadcl, nstrucdeadcl, nstablosscl, nstruclosscl, &
      distrdmleafcl,LeafAliveCl,FreeDMLeafCl, stressindexlvdyingcl, &
      RDyingLeafCl, LeafWtLossCl, LeafArLossCl,FrSenescence, &
      LeafArDeadCl, BlightStressIndexLvDyingCl
```

I.5. Subroutine GETCROPDATA

```
subroutine GetCropData(xNewTask, xModule, xLogFileUnit)
implicit none
include '..\GetCropData\CropDataSet.inc'
character*13 :: CropDataDone, EmptyString, OldTask
character*(*) :: xNewTask, xModule
integer :: xLogFileUnit
logical :: DoWaterStress, DoNitroStress, DoBlightStress, Dosoilblight
integer :: I, ifindc
SAVE
if ((xNewTask == 'initialize').or.(xNewTask == 'icinitialize')) then
   call Logmessage (xLogFileUnit, 9, xModule, '1.0')
   call getsrp (xModule, 'RGRL',RGRL)
   call getsrp (xModule, 'SLA' ,SLA)
   call getsrp (xModule, 'ECPDF' ,ECPDF)
   call getsrp (xModule, 'PARFraction',PARFraction)
   call getsrp (xModule, 'LUE',LUE)
   call getsrp (xModule, 'alphaT',alphaT)
   call getsrp (xModule, 'betaT',betaT)
   call getsrp (xModule, 'gammaT',gammaT)
   call getsrp (xModule, 'deltaT',deltaT)
   call getsrp (xModule, 'LeafPar',LeafPar)
   call getsrp (xModule, 'LeafStemRatio',LeafStemRatio)
   call getsrp (xModule, 'TuberFreshDryRatio',TuberFreshDryRatio)
   call getsrp (xModule, 'RootDepthCropMx', RootDepthCropMx)
   call getsrp (xModule, 'RootDepthGrowthPar', RootDepthGrowthPar)
   call getsl (xModule, 'DoWaterStress', DoWaterStress)
   call getsl (xModule, 'DoNitroStress', DoNitroStress)
   call getsl (xModule, 'DoBlightStress', DoBlightStress)
   call getsl (xModule, 'Dosoilblight', Dosoilblight)
   if (DoWaterStress) then
      call getsrp (xModule, 'KstrsRGRL',KstrsRGRL)
      call getsrp (xModule, 'KstrsAge',KstrsAge)
      call getsrp (xModule, 'KstrsLUE',KstrsLUE)
      call getsrp (xModule, 'Bage', Bage)
   end if
   call getarp (xModule, 'TMEffGrowthTB', TMEffGrowthTB, TMTableMxn, &
                TMEffGrowthTBL)
   call getsrp (xModule, 'TMBase',TMBase)
   if (DoNitroStress) then
      call getsrp (xModule, 'ksynstab', ksynstab)
      call getsrp (xmodule, 'kdestab', kdestab)
      call getsrp (xmodule, 'fractubstruc', fractubstruc)
      call getsrp (xmodule, 'fracabovestruc', fracabovestruc)
      call getsrp (xmodule, 'nstructconci', nstructconci)
      call getsrp (xmodule, 'nsolconci', nsolconci)
      call getsrp (xmodule, 'nstabconci', nstabconci)
   endif
   if (DoBlightStress.or.Dosoilblight) then
   endif
   call getac (xModule,'CNames', CNames, CNamesMxn, CNamesCnt)
   call getarp (xModule,'TMSumLeafGrowths', TMSumLeafGrowths, &
               CNamesMxn, i)
   call getarp (xModule,'LeafAgeMxs', LeafAgeMxs, CNamesMxn, i)
   call getarp (xModule,'LeafArGrowthRefs', LeafArGrowthRefs, &
               CNamesMxn, i)
   call getsc (xModule,'CName',CName)
   i = ifindc (CNames,CNamesMxn,1,CNamesCnt,CName)
   if (i > 0) then
      TMSumLeafGrowth = TMSumLeafGrowths(i)
      call putsrp (xModule, 'TMSumLeafGrowth', TMSumLeafGrowth)
      LeafAgeMx       = LeafAgeMxs(i)
      call putsrp (xModule, 'LeafAgeMx', LeafAgeMx)
      LeafArGrowthRef = LeafArGrowthRefs(i)
      call putsrp (xModule, 'LeafArGrowthRef', LeafArGrowthRef)
   else
```

```
      call fatalerr (xModule,'unknown cultivar')
   end if
   call getsrp (xModule, 'NPL', NPL)
   call getsi (xModule,'CropStYear'     ,CropStYear)
   call getsi (xModule,'CropStDay'     ,CropStDay)
   call getsrp  (xModule,'LeafArPlI'    ,LeafArPlI)
   call getsrp  (xModule,'RootDepthI'   ,RootDepthI)
   call getsrpm (xModule,'TMSumI'       ,TMSumI, 0.)
   call getsrpm (xModule,'LeafDeadWtI'  ,LeafDeadWtI, 0.)
   call getsrpm (xModule,'StemWtI'      ,StemWtI, 0.)
   call getsrpm (xModule,'RootWtI'      ,RootWtI, 0.)
   call getsrpm (xModule,'TuberWtI'     ,TuberWtI, 0.)
   call getsim  (xModule,'LeafClCntI'   ,LeafClCntI, 1)
   call getsrp (xModule,'WaterExtrPar',WaterExtrPar)
   call getsrp (xModule,'RainIntPar',RainIntPar)
end if
end subroutine GetCropData
```

I.6. Subroutine GETCROPDATALIST

```
subroutine GetCropDataList(xNewTask, xModule, xLogFileUnit)
implicit none
include '..\GetCropData\CropDataSet.inc'
character*13 :: CropDataDone, EmptyString, OldTask
character*(*) :: xNewTask, xModule
integer :: xLogFileUnit
logical :: DoWaterStress, DoNitroStress, DoBlightStress, Dosoilblight
integer :: i, cnamescnt2, ifindc
character*80 :: CropDataFile, NDemDataFile, TotalStressDataFile
character*30, dimension (65) :: ListRealParsPot
data ListRealParsPot /'TMSumI', 'LeafDeadWtI', 'StemWtI', 'TuberWtI', &
    'LUE', 'LeafArPlI', 'RootDepthI', 'PARFraction', 'ECPDF', &
    'ECPDFStem', 'TMBase', 'SLA', 'RGRL', 'LeafPar', 'LeafStemRatio', &
    'LfltAr', 'RelLossDeadLeaves', 'RelLossDeadStems', &
    'RelLossDeadTubers', 'RootDepthCropMx', 'RootDepthGrowthPar', &
    'AlphaT', 'BetaT', 'GammaT', 'DeltaT', 'TmsumLeafGrowth', &
    'LeafAgeMx', 'LeafArGrowthRef', 'GVILeaves', 'GVIStems', &
    'GVITubers','GVIReserves', 'RelRespTuber', 'ConvReserveDM', &
    'KminFint', 'KminScan', 'KmaxScan', 'betakminscan','betakmaxscan',&
    'akScan', 'bkScan', 'alphakScan', 'betakScan', 'alphaOWG', &
    'betaOWG', 'alphaUBG', 'betaUBG', 'alphaTuberFresh', &
    'betaTuberFresh', 'ksynstab', 'kdestabLeaf', 'kdestabStem', &
    'kdestabTuber', 'StVolLfAreaRatio', 'alphaLvAgeT', 'betaLvAgeT', &
    'frreserveleaf', 'frreservestem', 'frreservetuber', 'kAgeLue', &
    'bAgeLue', 'ecpdfmax', 'alphaecpdf', 'alphaStarch', 'betaStarch' /
real, dimension(65) :: ListDefValRealParsPot
data ListDefValRealParsPot / 5*0., 60*-99. /
integer :: NrRealParsPot = 65
character*30, dimension (7) :: ListRealParsH2O
data ListRealParsH2O / 'WaterExtrPar', 'RainIntPar', 'KStrsAge', &
    'BAge', 'GAge','KStrsAr','BetaStrsAr' /
real, dimension(7) :: ListDefValRealParsH2O
data ListDefValRealParsH2O / 7*-99. /
integer :: NrRealParsH2O = 7
character*30, dimension (43) :: ListRealParsNit
data ListRealParsNit / 'knLUE', 'noffset', 'knLvAg', 'bnLvAg', &
    'knStAg', 'bnStAg', 'knRGRL', 'bnRGRL', 'MaxSpad', 'alphaSpad', &
    'betaSpad', 'gammaSpad', 'deltaSpad', 'etaSpad', 'kappaSpad', &
    'NSolConcI', 'MaxNUprateMax', 'NStabLeafMax', 'NStabStemMax', &
    'NStabTuberMax','NStabLeafMin','NStabStemMin','NStabTuberMin', &
    'NStrucLeafMin','NStrucStemMin','NStrucTuberMin','DMNStabLeafMax',&
    'DMNStabStemMax', 'DMNStabTuberMax', 'DMNStabLeafMin', &
    'DMNStabStemMin', 'DMNStabTuberMin','DMNStrucLeaf','DMNStrucStem',&
    'DMNStrucTuber','alphaStabLeaf','betaStabLeaf','alphaStabStem', &
    'betaStabStem','alphaProtR','betaProtR','alphaProtT','betaProtT' /
real, dimension(43) :: ListDefValRealParsNit
data ListDefValRealParsNit / 43*-99. /
integer :: NrRealParsNit = 43
character*30, dimension (2) :: ListIntegerParsPot
data ListIntegerParsPot / 'LeafClCntI', 'nn' /
integer, dimension(2) :: ListDefValIntegerParsPot
data ListDefValIntegerParsPot / 1, -99 /
integer :: NrIntegerParsPot = 1
SAVE
if ((xNewTask == 'initialize').or.(xNewTask == 'icinitialize')) then
   call Logmessage (xLogFileUnit, 9, xModule, '1.0')
   call getsl (xModule, 'DoWaterStress', DoWaterStress)
   call getsl (xModule, 'DoNitroStress', DoNitroStress)
   call getsl (xModule, 'DoBlightStress', DoBlightStress)
   call getsl (xModule, 'Dosoilblight', Dosoilblight)
   call getscm (xModule, 'CropDataFile', CropDataFile, 'nn')
   if ((DoWaterStress) .or. (DoNitroStress)) then
      call getscm (xModule, 'TotalStressDataFile' , TotalStressDataFile, 'nn')
   end if
   if (DoNitroStress) then
```

```
      call getscm (xModule, 'NDemDataFile'        , NDemDataFile, 'nn')
   end if
   call loaddatafile (xModule, CropDataFile, xlogfileunit)
   do i = 1, nrrealparspot
      call SetRealVars ( xModule, ListRealParsPot(i), &
                         ListDefValRealParsPot(i))
   end do
   do i = 1, nrintegerparspot
      call SetIntegerVars ( xModule, ListIntegerParsPot(i), &
                         ListDefValIntegerParsPot(i))
   end do
   if ((DoWaterStress) .or. (DoNitroStress)) then
      call loaddatafile (xModule, TotalStressDataFile, xlogfileunit)
      do i = 1, nrrealparsH2O
         call SetRealVars ( xModule, ListRealParsH2O(i), &
                         ListDefValRealParsH2O(i))
      end do
      if (DoNitroStress) then
         call loaddatafile (xModule, NDemDataFile, xlogfileunit)
         do i = 1, nrrealparsH2O
            call SetRealVars ( xModule, ListRealParsNit(i), &
                         ListDefValRealParsNit(i))
         end do
      end if
   end if
end if
end subroutine GetCropDataList
```

I.7. Subroutine SETREALVARS

```
1   subroutine SetRealVars(xModule, VarName, VarDefVal)
2   implicit none
3   character*(*) :: VarName, xModule
4   real :: VarDefVal, xTemp
5   call getsrpm(xModule, VarName, xTemp, VarDefVal)
6   if ( ( abs(VarDefVal + 99.)<= 1.e-6) .and. ( abs(VarDefVal - xTemp)<= &
7      1.e-6)) then
8      call fatalerr(xModule, ' parameter ' // VarName // ' not defined, &
9                    pls check')
10  else
11     call cmdelvar (Varname)
12     call putsrp (xModule, VarName, xTemp)
13  end if
14  end subroutine SetRealVars
```

I.8. Subroutine SETINTEGERVARS

```
1   subroutine SetIntegerVars(xModule, VarName, VarDefVal)
2   implicit none
3   character*(*) :: VarName, xModule
4   integer :: VarDefVal, xTemp
5
6   call getsim (xModule, VarName, xTemp, VarDefVal)
7   if (abs(VarDefVal + 99)<= 0).and.( abs(VarDefVal - xTemp) == 0)) then
8      call fatalerr(xModule, ' parameter ' // VarName // ' not defined, &
9                    pls check')
10  else
11     call cmdelvar (Varname)
12     call putsi (xModule, VarName, xTemp)
13  end if
14  end subroutine SetIntegerVars
```

I.9. Subroutine GETCROPDATALIST_ARRAY

```
subroutine GetCropDataList_array(xNewTask, xMod, xLogFileUnit)
implicit none
include '..\GetCropData\CropDataSet.inc'
character*13 :: CropDataDone, EmptyString, OldTask
character*(*) :: xNewTask, xMod
character*80 :: CropDataFile, NDemDataFile, TotalStressDataFile
integer :: xLogFileUnit, i, cnamescnt2, ifindc
logical :: DoWaterStress, DoNitroStress, DoBlightStress, Dosoilblight
SAVE
if ((xNewTask == 'initialize').or.(xNewTask == 'icinitialize')) then
   call Logmessage (xLogFileUnit, 9, xMod, '1.0')
   call getsl (xMod, 'DoWaterStress', DoWaterStress)
   call getsl (xMod, 'DoNitroStress', DoNitroStress)
   call getsl (xMod, 'DoBlightStress', DoBlightStress)
   call getsl (xMod, 'Dosoilblight', Dosoilblight)
   call getscm (xMod, 'CropDataFile'           , CropDataFile, 'nn')
   if ((DoWaterStress) .or. (DoNitroStress)) then
      call getscm(xMod,'TotalStressDataFile',TotalStressDataFile, 'nn')
   end if
   if (DoNitroStress) then
      call getscm (xMod, 'NDemDataFile'        , NDemDataFile, 'nn')
   end if
   call getsc (xMod,'CName',CName)
   call lowerc(Cname)
   call cmdelvar('CNames')
   call cmdelvar('CNames2')
   call loaddatafile (xMod, CropDataFile, xlogfileunit)
   call getsrp (xMod, 'PARFraction',PARFraction)
   call getsrp  (xMod,'LeafArPlI'    ,LeafArPlI)
   call getsrp  (xMod,'RootDepthI'   ,RootDepthI)
   call getsrpm (xMod,'TMSumI'       ,TMSumI, 0.)
   call getsrpm (xMod,'LeafDeadWtI'  ,LeafDeadWtI, 0.)
   call getsrpm (xMod,'StemWtI'      ,StemWtI, 0.)
   call getsrpm (xMod,'RootWtI'      ,RootWtI, 0.)
   call getsrpm (xMod,'TuberWtI'     ,TuberWtI, 0.)
   call getsim  (xMod,'LeafClCntI'   ,LeafClCntI, 1)
   call cmdelvar ('LeafClCntI')
   call cmdelvar ('LeafArPlI')
   call cmdelvar ('RootDepthI')
   call cmdelvar ('TMSumI')
   call cmdelvar ('LeafDeadWtI')
   call cmdelvar ('StemWtI')
   call cmdelvar ('RootWtI')
   call cmdelvar ('TuberWtI')
   call putsrp (xMod,'LeafArPlI'    ,LeafArPlI)
   call putsrp (xMod,'RootDepthI'   ,RootDepthI)
   call putsrp (xMod,'TMSumI'       ,TMSumI)
   call putsrp (xMod,'LeafDeadWtI'  ,LeafDeadWtI)
   call putsrp (xMod,'StemWtI'      ,StemWtI)
   call putsrp (xMod,'RootWtI'      ,RootWtI)
   call putsrp (xMod,'TuberWtI'     ,TuberWtI)
   call putsi  (xMod,'LeafClCntI'   ,LeafClCntI)
   call getac (xMod,  'CNames',  CNames,  CNamesMxn, CNamesCnt)
   call getac (xMod,  'CNames2', CNames2, CNamesMxn, CNamesCnt2)
   if (CNamesCnt /= CNamesCnt2) then
      call fatalerr(xMod, 'Namelists in CROPDATAFILE are different')
   end if
   do i = 1, CNamesCnt
      call lowerc(Cnames(i))
   end do
   i = ifindc (CNames,CNamesMxn,1,CNamesCnt,CName)
   if (i > 0) then
      call choosevalue(CNamesCnt, i, 'LUEs',           'lue')
      call choosevalue(CNamesCnt, i, 'ECPDFs',         'ecpdf')
      call choosevalue(CNamesCnt, i, 'ECPDFStems',     'ecpdfstem')
```

```
66          call choosevalue(CNamesCnt, i, 'TMBases',        'tmbase')
67          call choosevalue(CNamesCnt, i, 'SLAs',           'slamax')
68          call choosevalue(CNamesCnt, i, 'RGRLs',          'rgrl')
69          call choosevalue(CNamesCnt, i, 'LeafPars',       'leafpar')
70          call choosevalue(CNamesCnt, i, 'LeafStemRatios', 'leafstemratio')
71          call choosevalue(CNamesCnt, i, 'LfltArs',         'lfltar')
72          call choosevalue(CNamesCnt, i, 'RelLossDeadLeavess', &
73                                         'rellossdeadleaves')
74          call choosevalue(CNamesCnt, i, 'RelLossDeadStemss',  &
75                                         'rellossdeadstems')
76          call choosevalue(CNamesCnt, i, 'RelLossDeadTuberss', &
77                                         'RelLossDeadTubers')
78          call choosevalue(CNamesCnt, i, 'RootDepthCropMxs', &
79                                         'RootDepthCropMx')
80          call choosevalue(CNamesCnt, i, 'RootDepthGrowthPars', &
81                                         'RootDepthGrowthPar')
82          call choosevalue(CNamesCnt, i, 'AlphaTs',        'alphaT')
83          call choosevalue(CNamesCnt, i, 'BetaTs',         'betaT')
84          call choosevalue(CNamesCnt, i, 'GammaTs',        'gammaT')
85          call choosevalue(CNamesCnt, i, 'DeltaTs',        'deltaT')
86          call choosevalue(CNamesCnt, i, 'TmsumLeafGrowths',    &
87                                         'TmsumLeafGrowth')
88          call choosevalue(CNamesCnt, i, 'LeafAgeMxs',     'LeafAgeMx')
89          call choosevalue(CNamesCnt, i, 'LeafArGrowthRefs', &
90                                         'LeafArGrowthRef')
91          call choosevalue(CNamesCnt, i, 'GVILeavess',     'GVILeaves')
92          call choosevalue(CNamesCnt, i, 'GVIStemss',      'GVIStems')
93          call choosevalue(CNamesCnt, i, 'GVITuberss',     'GVITubers')
94          call choosevalue(CNamesCnt, i, 'GVIReservess',   'GVIReserves')
95          call choosevalue(CNamesCnt, i, 'RelRespTubers',  'RelRespTuber')
96          call choosevalue(CNamesCnt, i, 'ConvReserveDMs', 'ConvReserveDM')
97          call choosevalue(CNamesCnt, i, 'KminFints',      'KminFint')
98          call choosevalue(CNamesCnt, i, 'KminScans',      'KminScan')
99          call choosevalue(CNamesCnt, i, 'KmaxScans',      'KmaxScan')
100         call choosevalue(CNamesCnt, i, 'betakminscans',  'betakminscan')
101         call choosevalue(CNamesCnt, i, 'betakmaxscans',  'betakmaxscan')
102         call choosevalue(CNamesCnt, i, 'akScans',        'akScan')
103         call choosevalue(CNamesCnt, i, 'bkScans',        'bkScan')
104         call choosevalue(CNamesCnt, i, 'alphakScans',    'alphakScan')
105         call choosevalue(CNamesCnt, i, 'betakScans',     'betakScan')
106         call choosevalue(CNamesCnt, i, 'alphaOWGs',      'alphaOWG')
107         call choosevalue(CNamesCnt, i, 'betaOWGs',       'betaOWG')
108         call choosevalue(CNamesCnt, i, 'alphaUBGs',      'alphaUBG')
109         call choosevalue(CNamesCnt, i, 'betaUBGs',       'betaUBG')
110         call choosevalue(CNamesCnt, i, 'alphaTuberFreshs', &
111                                        'alphaTuberFresh')
112         call choosevalue(CNamesCnt, i, 'betaTuberFreshs',  &
113                                        'betaTuberFresh')
114         call choosevalue(CNamesCnt, i, 'ksynstabs',      'ksynstab')
115         call choosevalue(CNamesCnt, i, 'kdestabLeafs',   'kdestabLeaf')
116         call choosevalue(CNamesCnt, i, 'kdestabStems',   'kdestabStem')
117         call choosevalue(CNamesCnt, i, 'kdestabTubers',  'kdestabTuber')
118         call choosevalue(CNamesCnt, i, 'kdestabRoots',   'kdestabRoot')
119         call choosevalue(CNamesCnt, i, 'StVolLfAreaRatios', &
120                                        'StVolLfAreaRatio')
121         call choosevalue(CNamesCnt, i, 'alphaLvAgeTs',   'alphaLvAgeT')
122         call choosevalue(CNamesCnt, i, 'betaLvAgeTs',    'betaLvAgeT')
123         call choosevalue(CNamesCnt, i, 'frreserveleafs', 'frreserveleaf')
124         call choosevalue(CNamesCnt, i, 'frreservestems', 'frreservestem')
125         call choosevalue(CNamesCnt, i, 'frreservetubers', &
126                                        'frreservetuber')
127         call choosevalue(CNamesCnt, i, 'kAgeLues',       'kAgeLue')
128         call choosevalue(CNamesCnt, i, 'bAgeLues',       'bAgeLue')
129         call choosevalue(CNamesCnt, i, 'ecpdfmaxs',      'ecpdfmax')
130         call choosevalue(CNamesCnt, i, 'alphaecpdfs',    'alphaecpdf')
131         call choosevalue(CNamesCnt, i, 'alphaStarchs',   'alphaStarch')
132         call choosevalue(CNamesCnt, i, 'betaStarchs',    'betaStarch')
133     else
```

```
134         call fatalerr (xMod,'unknown cultivar')
135      end if
136      call cmdelvar('CNames')
137      call cmdelvar('CNames2')
138      if ((DoWaterStress) .or. (DoNitroStress)) then
139         call loaddatafile (xMod, TotalStressDataFile, xlogfileunit)
140         call getac (xMod,  'CNames', CNames, CNamesMxn, CNamesCnt2)
141         if (CNamesCnt /= CNamesCnt2) then
142             call fatalerr(xMod, 'Namelists in TOTALSTRESSDATAFILE &
143                  are different from those in CROPDATAFILE')
144         end if
145         do i = 1, CNamesCnt
146            call lowerc(Cnames(i))
147         end do
148         i = ifindc (CNames,CNamesMxn,1,CNamesCnt,CName)
149         if (i > 0) then
150            call choosevalue(CNamesCnt, i, 'WaterExtrPars',
151                                           'WaterExtrPar')
152            call choosevalue(CNamesCnt, i, 'RainIntPars', 'RainIntPar')
153            call choosevalue(CNamesCnt, i, 'KStrsAges',   'KStrsAge')
154            call choosevalue(CNamesCnt, i, 'BAges',       'BAge')
155            call choosevalue(CNamesCnt, i, 'GAges',       'GAge')
156            call choosevalue(CNamesCnt, i, 'KStrsArs',    'KStrsAr')
157            call choosevalue(CNamesCnt, i, 'BetaStrsArs', 'BetaStrsAr')
158            if (DoNitroStress) then
159               call choosevalue(CNamesCnt, i, 'knLUEs',   'knLUE')
160               call choosevalue(CNamesCnt, i, 'noffsets', 'noffset')
161               call choosevalue(CNamesCnt, i, 'knLvAgs',  'knLvAg')
162               call choosevalue(CNamesCnt, i, 'bnLvAgs',  'bnLvAg')
163               call choosevalue(CNamesCnt, i, 'knStAgs',  'knStAg')
164               call choosevalue(CNamesCnt, i, 'bnStAgs',  'bnStAg')
165               call choosevalue(CNamesCnt, i, 'knRGRLs',  'knRGRL')
166               call choosevalue(CNamesCnt, i, 'bnRGRLs',  'bnRGRL')
167            end if
168         end if
169         call cmdelvar('CNames')
170         call cmdelvar('CNames2')
171         if (DoNitroStress) then
172            call loaddatafile (xMod, NDemDataFile, xlogfileunit)
173            call getac (xMod,  'CNames', CNames, CNamesMxn, CNamesCnt2)
174            if (CNamesCnt /= CNamesCnt2) then
175               call fatalerr(xMod, 'Namelists in NDEMDATAFILE are &
176                    different from those in CROPDATAFILE')
177            end if
178            call getac (xMod,'CNames2', CNames2, CNamesMxn, CNamesCnt2)
179            if (CNamesCnt /= CNamesCnt2) then
180               call fatalerr(xMod, 'Namelists in NDEMDATAFILE are &
181                    different from those in CROPDATAFILE')
182            end if
183            do i = 1, CNamesCnt
184               call lowerc(Cnames(i))
185            end do
186            i = ifindc (CNames,CNamesMxn,1,CNamesCnt,CName)
187            if (i > 0) then
188               call choosevalue(CNamesCnt, i, 'MaxSpads',    'MaxSpad')
189               call choosevalue(CNamesCnt, i, 'alphaSpads',  'alphaSpad')
190               call choosevalue(CNamesCnt, i, 'betaSpads',   'betaSpad')
191               call choosevalue(CNamesCnt, i, 'gammaSpads',  'gammaSpad')
192               call choosevalue(CNamesCnt, i, 'deltaSpads',  'deltaSpad')
193               call choosevalue(CNamesCnt, i, 'etaSpads',    'etaSpad')
194               call choosevalue(CNamesCnt, i, 'kappaSpads',  'kappaSpad')
195               call choosevalue(CNamesCnt, i, 'NSolConcIs',  'NSolConcI')
196               call choosevalue(CNamesCnt, i, 'MaxNUprateMaxs', &
197                                              'MaxNUprateMax')
198               call choosevalue(CNamesCnt, i, 'NStabLeafMaxs',  &
199                                              'NStabLeafMax')
200               call choosevalue(CNamesCnt, i, 'NStabStemMaxs',  &
201                                              'NStabStemMax')
```

```
202               call choosevalue(CNamesCnt, i, 'NStabRootMaxs',  &
203                                              'NStabRootMax')
204               call choosevalue(CNamesCnt, i, 'NStabTuberMaxs', &
205                                              'NStabTuberMax')
206               call choosevalue(CNamesCnt, i, 'NStabLeafMins', &
207                                              'NStabLeafMin')
208               call choosevalue(CNamesCnt, i, 'NStabStemMins', &
209                                              'NStabStemMin')
210               call choosevalue(CNamesCnt, i, 'NStabRootMins', &
211                                              'NStabRootMin')
212               call choosevalue(CNamesCnt, i, 'NStabTuberMins', &
213                                              'NStabTuberMin')
214               call choosevalue(CNamesCnt, i, 'NStrucLeafMins', &
215                                              'NStrucLeafMin')
216               call choosevalue(CNamesCnt, i, 'NStrucStemMins', &
217                                              'NStrucStemMin')
218               call choosevalue(CNamesCnt, i, 'NStrucRootMins', &
219                                              'NStrucRootMin')
220               call choosevalue(CNamesCnt, i, 'NStrucTuberMins', &
221                                              'NStrucTuberMin')
222               call choosevalue(CNamesCnt, i, 'DMNStabLeafMaxs', &
223                                              'DMNStabLeafMax')
224               call choosevalue(CNamesCnt, i, 'DMNStabStemMaxs', &
225                                              'DMNStabStemMax')
226               call choosevalue(CNamesCnt, i, 'DMNStabRootMaxs', &
227                                              'DMNStabRootMax')
228               call choosevalue(CNamesCnt, i, 'DMNStabTuberMaxs', &
229                                              'DMNStabTuberMax')
230               call choosevalue(CNamesCnt, i, 'DMNStabLeafMins', &
231                                              'DMNStabLeafMin')
232               call choosevalue(CNamesCnt, i, 'DMNStabStemMins', &
233                                              'DMNStabStemMin')
234               call choosevalue(CNamesCnt, i, 'DMNStabRootMins', &
235                                              'DMNStabRootMin')
236               call choosevalue(CNamesCnt, i, 'DMNStabTuberMins', &
237                                              'DMNStabTuberMin')
238               call choosevalue(CNamesCnt, i, 'DMNStrucLeafs',  &
239                                              'DMNStrucLeaf')
240               call choosevalue(CNamesCnt, i, 'DMNStrucStems',  &
241                                              'DMNStrucStem')
242               call choosevalue(CNamesCnt, i, 'DMNStrucRoots',  &
243                                              'DMNStrucRoot')
244               call choosevalue(CNamesCnt, i, 'DMNStrucTubers',  &
245                                              'DMNStrucTuber')
246               call choosevalue(CNamesCnt, i, 'alphaStabLeafs',  &
247                                              'alphaStabLeaf')
248               call choosevalue(CNamesCnt, i, 'betaStabLeafs',  &
249                                              'betaStabLeaf')
250               call choosevalue(CNamesCnt, i, 'alphaStabStems', &
251                                              'alphaStabStem')
252               call choosevalue(CNamesCnt, i, 'betaStabStems',  &
253                                              'betaStabStem')
254               call choosevalue(CNamesCnt, i, 'alphaProtRs', 'alphaProtR')
255               call choosevalue(CNamesCnt, i, 'betaProtRs',  'betaProtR')
256               call choosevalue(CNamesCnt, i, 'alphaProtTs', 'alphaProtT')
257               call choosevalue(CNamesCnt, i, 'betaProtTs',  'betaProtT')
258               call cmdelvar('kAgeLues')
259               call cmdelvar('bAgeLues')
260               call cmdelvar('frreserveleaves')
261               call cmdelvar('frreservestems')
262               call cmdelvar('frreservetubers')
263               call cmdelvar('slamax')
264               call cmdelvar('leafstemratio')
265               call cmdelvar('leafpar')
266               call cmdelvar('CNames')
267               call cmdelvar('CNames2')
268            else
269               call fatalerr (xMod,'unknown cultivar')
```

```
270             end if
271           end if
272        end if
273     end if
274     end subroutine GetCropDataList_array
```

I.10. Subroutine CHOOSEVALUE

```
1  subroutine choosevalue(arraysize, arrayelement, namearray, namevalue)
2  integer :: arraysize, arrayelement, i
3  character*(*) :: namearray, namevalue
4  real, dimension(arraysize) :: arrayx
5  call getarp ('ChooseValue', namearray,arrayx, arraysize, i)
6  call cmdelvar (namevalue)
7  call putsrp ('ChooseValue', namevalue, arrayx(arrayelement))
8  call cmdelvar (namearray)
9  end subroutine choosevalue
```

I.11. Subroutine GETSOILDATAPTFNEW

```
subroutine GetSoilDataPTFNew(xNewTask, xMod, xLogFileUnit)
implicit none
include '..\GetSoilData\SoilDataSet.inc'
integer :: i,ii, tmpi1, INl, inr, nrl, IERR, itmp1, xLogFileUnit
integer :: SoilNr, ISoilNr, NrHor
integer, parameter :: Idefault = -99, TempUnit = 50, MaxNrHor = 15
integer, dimension(MaxNrHor) :: DepthMin, DepthMax
integer, dimension(MaxNrHor) :: lutum, leem, m50, moedermat
character*13 :: SoilDataDone, EmptyString, Oldtask
character*(*) :: xNewTask, xMod
character*80 :: SoilDataBase
character*1 :: fdum, LandUse
character*5 :: HorName
character*6 :: SoilUnit
real, parameter :: Default = -99.
real, dimension(MaxNrHor) :: VGWRh, VGWSh, VGKSh, VGAh, VGNh, VGLh
real, dimension(MaxNrHor) :: OrgMat, pHKCL, pHH2O, density
real, dimension(nlmxn) :: PERCH, PERNH, FSOM, RHODH
real :: OrgMatSoil, OrgNMatSoil, FrCinOrgMatSoil, StandCNOrgMatSoil
real :: MaxCNorgMatSoil, MinCNorgMatSoil, alphaCNOrgMatSoil, CNLOM
real :: MaxThickL, depth, PloughDepth, FLOM0, alphaFSOM
SAVE
if (xNewTask == 'initialize') then
   call Logmessage (xLogFileUnit, 9, xMod, '1.0')
   do i = 1,nlmxn
      VGR(i)   = 0.
      WCST(i)  = 0.
      KST(i)   = 0.
      VGA(i)   = 0.
      VGN(i)   = 0.
      VGL(i)   = 0.
      RHOD(i)  = 0.
      PerC(i)  = 0.
      PerN(i)  = 0.
      wclqt(i) = 0.
      WCFC(i)  = 0.
      WCWP(i)  = 0.
      WCAD(i)  = 0.
      WCWU(i)  = 0.
      WCWO(i)  = 0.
      SoilTp(i)= 0.
      CLOM(i)  = 0.
      CSOM(i)  = 0.
      CDPM(i)  = 0.
      CSPM(i)  = 0.
      CRPM(i)  = 0.
      ANLAY(i) = 0.
      NSOM(i)  = 0.
      QAFERonSoil(i) = 0.
   end do
   call getsim (xMod, 'swit9', swit9, IDefault)
   call getsim (xMod, 'swit8', swit8, IDefault)
   call getsim (xMod, 'swit7', swit7, IDefault)
   call getsim (xMod, 'swit6', swit6, IDefault)
   call getsim (xMod, 'swit5', swit5, IDefault)
   call getsim (xMod, 'swit3', swit3, IDefault)
   IF (SWIT9 == 2 .AND. ((SWIT3 == 1 .AND. SWIT8 /= 1) .OR. &
    SWIT3 == 2 .AND. SWIT8 /= 1).OR.(SWIT3 == 3 .AND. SWIT8 /= 2).OR.&
   (SWIT3 /= 1 .AND. SWIT3 /= 2 .AND. SWIT3 /= 3 .AND. SWIT3 /= &
   IDefault))) CALL fatalerr(xMod,'switch inconsistency SWIT3/8/9')
   call getsim  (xMod,'Nl',Nl,0)
   call getarpm (xMod,'tkl',tkl,NlMxn,itmp1,0.)
   call getsrpm (xMod, 'DTMIN',DTMIN, Default)
   CALL getsrpm (xMod, 'DTMX1',DTMX1, Default)
   call getsrpm (xMod, 'WATUP', WATUP, FIELD)
```

```
      if (watup > AirDr) watup = airdr - tiny4
      call getsrpm (xMod, 'WATOX', WATOX, FIELD/2.)
      if (watox <= 0.) watox = tiny4
      call cmdelvar('watox')
      call putsrp(xMod, 'watox', watox)
      IF (SWIT5 == 2) call getsrpm (xMod,'DTFX',DTFX, Default)
      IF (SWIT9 == 1) THEN
         CALL fatalerr(xMod,'swit9 can not be 1 if using soil database')
      ELSE IF (SWIT9 == 2) THEN
         call getsrpm (xMod, 'OrgMatSoil', OrgMatSoil, Default)
         call getsrpm (xMod, 'OrgNMatSoil', OrgNMatSoil, Default)
         call getsrpm (xMod, 'MaxCNorgMatSoil', MaxCNorgMatSoil, 150.)
         call getsrpm (xMod, 'MinCNorgMatSoil', MinCNorgMatSoil, 6.)
         call getsrpm (xMod, 'alphaCNOrgMatSoil',alphaCNOrgMatSoil, 2.)
         call getsrpm (xMod, 'FLOM0',FLOM0, 0.9)
         call getsrpm (xMod, 'alphaFSOM',alphaFSOM, 0.1)
         call getsrpm (xMod, 'FrCinOrgMatSoil', FrCinOrgMatSoil, 0.58)
         call getsrpm (xMod, 'PloughDepth', PloughDepth, 0.4)
         call getsrpm (xMod, 'CNLOM',CNLOM, 15.)
         call getsi (xMod, 'SoilNr',SoilNr)
         call getsc (xMod, 'SoilDataBase', SoilDataBase)
         open (unit=TempUnit, file=SoilDataBase, status = 'old')
1        read (TempUnit,*, end=3) ISoilNr, SoilUnit, LandUse
         read (TempUnit,*) NrHor
         if (IsoilNr == SoilNr) then
            do i = 1, NrHor
               read(TempUnit, *) HorName, DepthMin(i), DepthMax(i), &
                    OrgMat(i), Lutum(i), Leem(i), M50(i), pHKCL(i), &
                    pHH2O(i), moedermat(i), density(i), VGWRh(i),   &
                    VGWSh(i), VGKSh(i),VGAh(i),VGLh(i),VGNh(i)
            end do
            close(TempUnit)
            goto 2
         else
            do i = 1,NrHor
               read (TempUnit,*) fdum
            end do
            goto 1
         endif
3        CALL fatalerr(xMod,' SoilNr not in file ' // SoilDataBase)
2        continue
         call putarp (xMod, 'DepthMin', DepthMin/100., NrHor)
         call putarp (xMod, 'DepthMax', DepthMax/100., NrHor)
         call putarp (xMod, 'OrgMat', OrgMat, NrHor)
         call putai  (xMod, 'Lutum', Lutum, NrHor)
         call putai  (xMod, 'Leem', Leem, NrHor)
         call putai  (xMod, 'M50', M50, NrHor)
         call putarp (xMod, 'density', density, NrHor)
         call putai  (xMod, 'MoederMat', MoederMat, NrHor)
      IF (SWIT3 /= 3) THEN
         CALL fatalerr(xMod,'swit3 should be 3 for soildatabase')
      ELSE IF (SWIT3 == 3) THEN
         if (itmp1 == 0) then
            if (Nl == 0) then
               MaxThickL = max(10., DepthMax(NrHor) * 1. / NlMxn)
               Do i = 1, NrHor
                  inr = nint((DepthMax(i) - DepthMin(i))/MaxThickL)
                  depth = DepthMin(i)/100.
                  do ii = Nl+1,Nl + inr
                     tkl(ii) = min(DepthMax(i) - Depth*100., &
                               1.*(DepthMax(i)-DepthMin(i))/inr) / 100.
                     Depth = Depth + tkl(ii)
                     VGR(ii)  = VGWRh(I)
                     WCST(ii) = VGWSh(I)
                     KST(ii)  = VGKSh(I)
                     VGA(ii)  = VGAh(I)
                     VGN(ii)  = max(1.+tiny2,VGNh(I))
                     VGL(ii)  = VGLh(I)
```

```
                  RHODH(ii) = density(i)
                  if((OrgMatSoil >= 0.) .and. (PloughDepth - Depth &
                     >= -0.5*tkl(ii))) then
                     PerCH(ii) = OrgMatSoil * FrCinOrgMatSoil
                  else
                     PerCH(ii) = OrgMat(i) * FrCinOrgMatSoil
                  endif
                  if((OrgNMatSoil >= 0.).and. (PloughDepth - Depth &
                     >= -0.5*tkl(ii))) then
                     PerNH(ii) = OrgNMatSoil
                     FSOM(ii)  = 1.- min(0.999, max(0.005, FLOM0 * &
                     exp(max(-30., min(30.,-alphaFSOM*PerCH(ii))))))
                  else
                     StandCNOrgMatSoil = MinCNOrgMatSoil + &
                       (MaxCNOrgMatSoil - MinCNOrgMatSoil) * &
                       (1. - exp(max(-30.,min(30., - PerCH(ii) * &
                        alphaCNOrgMatSoil /100.))))
                     PerNH(ii) = PerCH(ii) / StandCNOrgMatSoil
                     FSOM(ii)  = 1.- min(0.999, max(0.005, FLOM0 * &
                     exp(max(-30., min(30.,-alphaFSOM*PerCH(ii))))))
                  endif
               end do
               Nl = Nl + inr
            end do
         else
            INl = 0
            do i = 1, NrHor
               if ((DepthMax(i) - DepthMin(i) >= 15.) .and. &
                  (DepthMin(i) <= 50.)) then
                   nrl = nint((depthmax(i)-Depthmin(i))/9.5)
                   do ii = 1, nrl
                      tkl(inl+ii) = (DepthMax(i)-DepthMin(i)) / &
                                    (100. * nrl)
                   end do
                   INl = INl + nrl
               elseif ((DepthMax(i) - DepthMin(i) >= 20.) .and. &
                  (DepthMin(i) <= 100.)) then
                   nrl = nint((depthmax(i)-Depthmin(i))/15.)
                   do ii = 1, nrl
                      tkl(inl+ii) = (DepthMax(i)-DepthMin(i)) / &
                                    (100. * nrl)
                   end do
                   INl = INl + nrl
               else
                  INl = INl + 1
                  tkl(inl) = (DepthMax(i)-DepthMin(i))/100.
               end if
            end do
            if (INl > Nl) then
               Nl = INl
            end if
            depth = 0.
            i = 1
            do ii = 1, Nl
               depth = depth + tkl(ii) * 100.
               if (DepthMax(i) + 1.e-3 < depth) then
                  i = i + 1
               end if
               if (i > NrHor) then
                  call fatalerr(xMod, ' depth of layer larger than &
                       depth of horizons')
               end if
               VGR(ii)  = VGWRh(I)
               WCST(ii) = VGWSh(I)
               KST(ii)  = VGKSh(I)
               VGA(ii)  = VGAh(I)
               VGN(ii)  = max(1.+tiny2, VGNh(I))
               VGL(ii)  = VGLh(I)
```

```
                  RHODH(ii) = density(i)
                  if ((OrgMatSoil >= 0.).and. (PloughDepth - Depth >= &
                     -0.5*tkl(ii))) then
                     PerCH(ii) = OrgMatSoil * FrCinOrgMatSoil
                  else
                     PerCH(ii) = OrgMat(i) * FrCinOrgMatSoil
                  endif
                  if ((OrgNMatSoil >= 0.).and.(PloughDepth - Depth >= &
                     -0.5*tkl(ii))) then
                     PerNH(ii) = OrgNMatSoil
                     FSOM(ii)  = 1. - min(0.999, max(0.005, FLOM0 * &
                     exp(max(-30., min(30., -alphaFSOM * PerCH(ii))))))
                  else
                     StandCNOrgMatSoil = MinCNOrgMatSoil + &
                     (MaxCNOrgMatSoil - MinCNOrgMatSoil) * &
                     (1. - exp(max(-30.,min(30., -alphaCNOrgMatSoil &
                          * PerCH(ii)/100.))))
                     PerNH(ii) = PerCH(ii) / StandCNOrgMatSoil
                     FSOM(ii)  = 1. - min(0.999, max(0.005, FLOM0 * &
                     exp(max(-30., min(30., -alphaFSOM * PerCH(ii))))))
                  endif
               End Do
               if (INl < Nl ) then
                  if (abs(DepthMax(NrHor) - Depth )> 1.e-3) then
                     do ii = INl +1, Nl
                        tkl(ii)   = (DepthMax(NrHor) - Depth) / &
                                    (100.*(Nl - INl))
                        VGR(ii)   = VGR(INl)
                        WCST(ii)  = WCST(INl)
                        KST(ii)   = KST(INl)
                        VGA(ii)   = VGA(INl)
                        VGN(ii)   = VGN(INl)
                        VGL(ii)   = VGL(INl)
                        RHODH(ii) = rhodh(INl)
                        PerCH(ii) = PerCH(INl)
                        PerNH(ii) = PerNH(INl)
                        FSOM(ii)  = FSOM(INl)
                     end do
                  else
                     ii = 0
554                  ii = ii + 1
                     if (ii <= INl) then
                        if (tkl(ii) >= 0.175) then
                           do i = inl, ii+1, -1
                              tkl(i+1)   = tkl(i)
                              VGR(i+1)   = VGR(i)
                              WCST(i+1)  = WCST(i)
                              KST(i+1)   = KST(i)
                              VGA(i+1)   = VGA(i)
                              VGN(i+1)   = VGN(i)
                              VGL(i+1)   = VGL(i)
                              RHODH(i+1) = rhodh(i)
                              PerCH(i+1) = PerCH(i)
                              PerNH(i+1) = PerNH(i)
                              FSOM(i+1) = FSOM(i)
                           end do
                           tkl(ii+1)   = tkl(ii)/2.
                           tkl(ii)     = tkl(ii)/2.
                           VGR(ii+1)   = VGR(ii)
                           WCST(ii+1)  = WCST(ii)
                           KST(ii+1)   = KST(ii)
                           VGA(ii+1)   = VGA(ii)
                           VGN(ii+1)   = VGN(ii)
                           VGL(ii+1)   = VGL(ii)
                           RHODH(ii+1) = rhodh(ii)
                           PerCH(ii+1) = PerCH(ii)
                           PerNH(ii+1) = PerNH(ii)
                           FSOM(ii+1) = FSOM(ii)
```

```
                              INl = INl + 1
                              if (INl == Nl) goto 555
                           end if
                           goto 554
                        end if
                        Nl = INl
                     end if
                  end if
               end if
555            continue
            elseif ((itmp1 == Nl) .or. (Nl == 0)) then
               Nl = itmp1
               depth = 0.
               ii = 1
               i = 1
10             depth = depth + 100.*tkl(ii)
20             if ((depth - DepthMin(i) > -tiny3) .and. (depth - &
                   DepthMax(i) < tiny3)) then
                  VGR(ii)  = VGWRh(I)
                  WCST(ii) = VGWSh(I)
                  KST(ii)  = VGKSh(I)
                  VGA(ii)  = VGAh(I)
                  VGN(ii)  = max(1.+tiny2, VGNh(I))
                  VGL(ii)  = VGLh(I)
                  RHODH(ii) = density(i)
                  if ((OrgMatSoil >= 0.) .and. (PloughDepth - Depth >= &
                      -0.5*tkl(ii))) then
                     PerCH(ii) = OrgMatSoil * FrCinOrgMatSoil
                  else
                     PerCH(ii) = OrgMat(i) * FrCinOrgMatSoil
                  endif
                  if ((OrgNMatSoil >= 0.) .and. (PloughDepth - Depth >= &
                      -0.5*tkl(ii))) then
                     PerNH(ii) = OrgNMatSoil
                     FSOM(ii)  = 1. - min(0.999, max(0.005, FLOM0 * &
                        exp(max(-30., min(30., -alphaFSOM * PerCH(ii))))))
                  else
                     StandCNOrgMatSoil = MinCNOrgMatSoil + &
                        (MaxCNOrgMatSoil - MinCNOrgMatSoil) * &
                        (1. - exp(max(-30.,min(30., -alphaCNOrgMatSoil * &
                         PerCH(ii)/100.))))
                     PerNH(ii) = PerCH(ii) / StandCNOrgMatSoil
                     FSOM(ii)  = 1. - min(0.999, max(0.005, FLOM0 * &
                        exp(max(-30., min(30., -alphaFSOM * PerCH(ii))))))
                  endif
                  ii = ii + 1
                  if (ii == Nl + 1) GOTO 30
                  goto 10
               elseif (depth - depthmax(i) > -1.e-3 ) then
                  i = i + 1
                  if (i > NrHor) then
                     CALL fatalerr(xMod,' Given TKL leads to deeper &
                        profile than given in file ' // SoilDataBase)
                  else
                     goto 20
                  end if
               elseif (depth - depthmin(i) < -tiny3) then
                  CALL fatalerr(xMod,' Given TKL does not correspond to &
                     profile given in file ' // SoilDataBase)
               endif
30             continue
            else
               CALL fatalerr(xMod,'error in number of layers; compare Nl &
                  and dimension of TKL')
            end if
         end if
      ELSE
         call fatalerr (xMod,'SWIT9 wrong value ; should be 2')
```

```
      END IF
      DO I = 1,NL
         CALL SUWCMS (xLogFileUnit,I,2,WCFC(I),FIELD)
         CALL SUWCMS (xLogFileUnit,I,2,WCWP(I),WILTP)
         CALL SUWCMS (xLogFileUnit,I,2,WCAD(I),AIRDR)
         CALL SUWCMS (xLogFileUnit,I,2,WCWU(I),WatUp)
         CALL SUWCMS (xLogFileUnit,I,2,WCWO(I),WatOx)
         if(wcfc(i) > wcst(i)-tiny4) then
            wcfc(i) = wcst(i)-tiny4
         endif
         if(wcwp(i) > wcfc(i)-tiny4) then
            wcwp(i) = wcfc(i)-tiny4
         endif
         if(wcad(i) > wcwp(i)-tiny4) then
            wcad(i) = wcwp(i)-tiny4
         endif
          if(wcwu(i) > wcst(i)-tiny4) then
            wcwu(i) = wcst(i)-tiny4
         endif
          if(wcwo(i) > wcst(i)-tiny4) then
            wcwo(i) = wcst(i)-tiny4
         endif
!        Calculate array with depths of middle of each layer
         if (i == 1) then
            ULimit(i) = 0.
            LLimit(i) = TKL(i)
         else
            ULimit(i) = LLimit(i-1)
            LLimit(i) = ULimit(i) + TKL(i)
         endif
         if (i ==1) then
            exposedlayer(i) = 1.
            frexposedlayer(i) = 1.
         else
            exposedlayer(i) = 0.
            frexposedlayer(i) = 0.
         end if
         frvolumelayer(i) = 1.
      end do
      Nl = Nl + 2
      nlexp = 3
      istart = 3
      do i = Nl, 3, -1
         tkl(i)   = tkl(i-2)
         VGR(i)   = vgr(i-2)
         WCST(i)  = wcst(i-2)
         KST(i)   = kst(i-2)
         VGA(i)   = vga(i-2)
         VGN(i)   = vgn(i-2)
         VGL(i)   = vgl(i-2)
         RHODH(i) = rhodh(i-2)
         PerCH(i) = perch(i-2)
         PerNH(i) = pernh(i-2)
         FSOM(i)  = FSOM(i-2)
         wcfc(i)  = wcfc(i-2)
         wcad(i)  = wcad(i-2)
         wcwp(i)  = wcwp(i-2)
         wcwu(i)  = wcwu(i-2)
         wcwo(i)  = wcwo(i-2)
         exposedlayer(i) = exposedlayer(i-2)
         frexposedlayer(i) = frexposedlayer(i-2)
         frvolumelayer(i) = frvolumelayer(i-2)
         ulimit(i) = ulimit(i-2)
         llimit(i) = llimit(i-2)
      end do
      do i = 1, 2
         tkl(i)   = 0.
         VGR(i)   = 0.
```

```
406         WCST(i)  = 0.
407         KST(i)   = 0.
408         VGA(i)   = 0.
409         VGN(i)   = 0.
410         VGL(i)   = 0.
411         RHODH(i) = 0.
412         PerCH(i) = 0.
413         PerNH(i) = 0.
414         FSOM(i)  = 0.
415         wcfc(i)  = 0.
416         wcad(i)  = 0.
417         wcwp(i)  = 0.
418         wcwu(i)  = 0.
419         wcwo(i)  = 0.
420         exposedlayer(i) = 0.
421         frexposedlayer(i) = 0.
422         frvolumelayer(i) = 0.
423         ulimit(i) = 0.
424         llimit(i) = 0.
425       end do
426       call cmdelvar( 'Nl')
427       call putsi (xMod, 'Nl', Nl)
428       call cmdelvar( 'Nlexp')
429       call putsi (xMod, 'Nlexp', Nlexp)
430       call cmdelvar( 'ulimit')
431       call cmdelvar( 'llimit')
432       call cmdelvar( 'tkl')
433       call cmdelvar( 'vgr')
434       call cmdelvar( 'kst')
435       call cmdelvar( 'vga')
436       call cmdelvar( 'vgl')
437       call cmdelvar( 'vgn')
438       call putarp (xMod, 'vgr', vgr, Nl)
439       call putarp (xMod, 'kst', kst, Nl)
440       call putarp (xMod, 'vga', vga, Nl)
441       call putarp (xMod, 'vgn', vgn, Nl)
442       call putarp (xMod, 'vgl', vgl, Nl)
443       call cmdelvar( 'wcfc')
444       call cmdelvar( 'wcwp')
445       call cmdelvar( 'wcad')
446       call cmdelvar( 'wcst')
447       call cmdelvar( 'wcwu')
448       call cmdelvar( 'wcwo')
449       call putarp (xMod, 'wcfc', wcfc, Nl)
450       call putarp (xMod, 'wcwp', wcwp, Nl)
451       call putarp (xMod, 'wcad', wcad, Nl)
452       call putarp (xMod, 'wcst', wcst, Nl)
453       call putarp (xMod, 'wcwu', wcwu, Nl)
454       call putarp (xMod, 'wcwo', wcwo, Nl)
455       call cmdelvar( 'rhodh')
456       call cmdelvar( 'perch')
457       call cmdelvar( 'pernh')
458       call cmdelvar( 'fsom')
459       call putarp (xMod, 'rhodh', rhodh, Nl)
460       call putarp (xMod, 'perch', perch, Nl)
461       call putarp (xMod, 'pernh', pernh, Nl)
462       call putarp (xMod, 'fsom', fsom, Nl)
463       call getsrpm (xMod, 'CSC2  ',CSC2, Default )
464       call getsrpm (xMod, 'CSA   ',CSA,  Default  )
465       call getsrpm (xMod, 'CSB   ',CSB,  Default  )
466       call getsrpm (xMod, 'WL0MX ',WL0MX, Default )
467       call getsim (xMod,'IDRAIN',IDRAIN, IDefault)
468       call getarpm (xMod, 'ZWTB',ZWTB,ILZMAX,IZWTB,Default)
469       call getsrpm (xMod,'ees',ees, Default)
470       call getsrpm (xMod,'ro1',ro1, Default)
471       call getsrpm (xMod,'ro2',ro2, Default)
472       SoilDataDone = 'Ready'
473     end if
474     end subroutine GetSoilDataPTFNew
```

I.12. Subroutine PEDOTRANS_VGN

```
Subroutine PedoTrans_VGN(xLogFileUnit, pldep)
! Based on: J.H.M. Wösten, G.J. Veerman, W.J.M. de Groot & J. Stolte,
! 2001. Waterretentie- en doorlatendheidskarakteristieken van boven- en
! ondergronden in Nederland: de Staringreeks Vernieuwde uitgave 2001.
! Alterra-rapport 153, Alterra, WUR, Wageningen, 86 pp.
implicit none
include '..\GetSoilData\SoilDataSet.inc'
logical :: oligotroof
character(30), parameter :: xMod = 'Pedotrans_VGN'
real, dimension(50) :: pGenuchtenClay, pGenuchtenSand
real :: wcstx, kstx, aVGNx, lVGNx, nVGNx, OrgMatTot, LutumTot, LeemTot
real, dimension(NlMxn) :: DepthMin, DepthMax, OrgMat, density, frHor
real :: WCFCX, WCWPX, WCADX, WCWUX, WCWOX, M50Tot, densityTot, Xtot
real :: XOligotroof, pldep, X, XBovenGrond, Bovengrond, dichtheid
integer :: npGenuchtenClay, npGenuchtenSand, npClay, npSand, NrHor
integer :: HorLow, j, i, xLogFileUnit
integer, dimension(NlMxn) :: moedermat, Lutum, Leem, M50
call getsi (xMod, 'npGenuchtenClay', npGenuchtenClay)
call getsi (xMod, 'npGenuchtenSand', npGenuchtenSand)
call getarp (xMod, 'pGenuchtenClay', pGenuchtenClay , &
                    npGenuchtenClay, npClay)
call getarp (xMod, 'pGenuchtenSand', pGenuchtenSand , &
                    npGenuchtenSand, npSand)
if (npClay /= npGenuchtenClay) then
   call fatalerr(xMod, 'incorrect nr Pedotransfer parameters for Clay')
end if
if (npSand /= npGenuchtenSand) then
   call fatalerr(xMod, 'incorrect nr Pedotransfer parameters for Sand')
end if
call getarp  (xMod, 'DepthMin'  , DepthMin  , NlMxn, NrHor)
call getarp  (xMod, 'DepthMax'  , DepthMax  , NlMxn, NrHor)
call getarp  (xMod, 'OrgMat'    , OrgMat    , NlMxn, NrHor)
call getai   (xMod, 'Lutum'     , Lutum     , NlMxn, NrHor)
call getai   (xMod, 'Leem'      , Leem      , NlMxn, NrHor)
call getai   (xMod, 'M50'       , M50       , NlMxn, NrHor)
call getai   (xMod, 'moedermat' , moedermat , NlMxn, NrHor)
call getarp  (xMod, 'density'   , density   , NlMxn, NrHor)
HorLow = 0
do j = 1, NrHor
   frHor(j) = 1.
   if (DepthMax(j)  >= pldep) then
      HorLow = j
      frHor(j) = (PlDep - DepthMin(j) ) / (DepthMax(j) - DepthMin(j))
      goto 555
   end if
end do
555 OrgMatTot  = 0.
LutumTot   = 0.
LeemTot    = 0.
M50Tot     = 0.
densityTot = 0.
XTot       = 0.
XOligotroof = 0.
oligotroof = .false.
Xbovengrond = 0.
Bovengrond = 0.
do j = 1, HorLow
   X = frHor(j) * (DepthMax(j) - DepthMin(j)) * density(j)
   XTot = XTot + X
   if ((moedermat(j) >= 150) .and. (moedermat(j) <= 152)) then
      xoligotroof = xoligotroof + X
   end if
   if (DepthMax(j) <= 0.40) then
      XBovengrond = XBovengrond + X
   end if
```

```
      OrgMatTot  = OrgMatTot + OrgMat(j) * X
      LutumTot   = LutumTot + Lutum(j) * X
      LeemTot    = LeemTot + Leem(j) * X
      M50Tot     = M50Tot + M50(j) * X
      densityTot = densityTot + density(j) * X
   end do
   if (XOligotroof / Xtot > 0.5) then
      oligotroof = .true.
   end if
   if (XBovengrond / XTot > 0.5) then
      Bovengrond = 1.
   end if
   OrgMatTot  = OrgMatTot / XTot
   LutumTot   = LutumTot / XTot
   LeemTot    = LeemTot / XTot
   M50Tot     = M50Tot / XTot
   densityTot = densityTot / XTot
   if (bovengrond >= 0.9) then
      if (OrgMatTot <= 15.) then
         if ((LeemTot < 50.) .and. (LutumTot < 8.0)) then
            call GetNewVanGenuchtenParamsSand(npGenuchtenSand, &
               pGenuchtenSand, M50Tot, LeemTot, OrgMatTot, bovengrond, &
               dichtheid, wcstX, kstX, aVGNX, lVGNX, nVGNX)
         else if ((LutumTot >= 8.0) .and. (LeemTot < 50.0)) then
            call GetNewVanGenuchtenParamsClay(npGenuchtenClay, &
               pGenuchtenClay, LutumTot, OrgMatTot, dichtheid, wcstX, &
               kstX, aVGNX, lVGNX, nVGNX)
         elseif (LeemTot < 85.) then
            call getsrp (xMod, 'wcst_B13', wcstX)
            call getsrp (xMod, 'wcst_B13', kstX)
            call getsrp (xMod, 'aVGN_B13', aVGNX)
            call getsrp (xMod, 'lVGN_B13', lVGNX)
            call getsrp (xMod, 'nVGN_B13', nVGNX)
         else
            call getsrp (xMod, 'wcst_B14', wcstX)
            call getsrp (xMod, 'wcst_B14', kstX)
            call getsrp (xMod, 'aVGN_B14', aVGNX)
            call getsrp (xMod, 'lVGN_B14', lVGNX)
            call getsrp (xMod, 'nVGN_B14', nVGNX)
         end if
      else if (LutumTot <= 8.) then
         if (OrgMatTot <= 25.) then
            call getsrp (xMod, 'wcst_B15', wcstX)
            call getsrp (xMod, 'wcst_B15', kstX)
            call getsrp (xMod, 'aVGN_B15', aVGNX)
            call getsrp (xMod, 'lVGN_B15', lVGNX)
            call getsrp (xMod, 'nVGN_B15', nVGNX)
         else
            call getsrp (xMod, 'wcst_B16', wcstX)
            call getsrp (xMod, 'wcst_B16', kstX)
            call getsrp (xMod, 'aVGN_B16', aVGNX)
            call getsrp (xMod, 'lVGN_B16', lVGNX)
            call getsrp (xMod, 'nVGN_B16', nVGNX)
         end if
      elseif(OrgMatTot > 15.) then
         if (OrgMatTot <= 25.) then
            call getsrp (xMod, 'wcst_B17', wcstX)
            call getsrp (xMod, 'wcst_B17', kstX)
            call getsrp (xMod, 'aVGN_B17', aVGNX)
            call getsrp (xMod, 'lVGN_B17', lVGNX)
            call getsrp (xMod, 'nVGN_B17', nVGNX)
         else
            call getsrp (xMod, 'wcst_B18', wcstX)
            call getsrp (xMod, 'wcst_B18', kstX)
            call getsrp (xMod, 'aVGN_B18', aVGNX)
            call getsrp (xMod, 'lVGN_B18', lVGNX)
            call getsrp (xMod, 'nVGN_B18', nVGNX)
         end if
```

```
      else
      end if
   else
      if (OrgMatTot <= 3.) then
         if ((LeemTot < 50.) .and. (LutumTot < 8.0)) then
            call GetNewVanGenuchtenParamsSand(npGenuchtenSand, &
               pGenuchtenSand, M50Tot, LeemTot, OrgMatTot, bovengrond, &
               dichtheid, wcstX, kstX, aVGNX, lVGNX, nVGNX)
         else if ((LutumTot >= 8.0) .and. (LeemTot < 50.0)) then
            call GetNewVanGenuchtenParamsClay(npGenuchtenClay, &
               pGenuchtenClay, LutumTot, OrgMatTot, &
               dichtheid, wcstX, kstX, aVGNX, lVGNX, nVGNX)
         elseif (LeemTot < 85.) then
            call getsrp (xMod, 'wcst_O14', wcstX)
            call getsrp (xMod, 'wcst_O14', kstX)
            call getsrp (xMod, 'aVGN_O14', aVGNX)
            call getsrp (xMod, 'lVGN_O14', lVGNX)
            call getsrp (xMod, 'nVGN_O14', nVGNX)
         else
            call getsrp (xMod, 'wcst_O15', wcstX)
            call getsrp (xMod, 'wcst_O15', kstX)
            call getsrp (xMod, 'aVGN_O15', aVGNX)
            call getsrp (xMod, 'lVGN_O15', lVGNX)
            call getsrp (xMod, 'nVGN_O15', nVGNX)
         end if
      elseif (OrgMatTot > 3.) then
         if (OrgMatTot <= 35.) then
            call getsrp (xMod, 'wcst_O18', wcstX)
            call getsrp (xMod, 'wcst_O18', kstX)
            call getsrp (xMod, 'aVGN_O18', aVGNX)
            call getsrp (xMod, 'lVGN_O18', lVGNX)
            call getsrp (xMod, 'nVGN_O18', nVGNX)
         elseif (not(oligotroof)) then
            call getsrp (xMod, 'wcst_O17', wcstX)
            call getsrp (xMod, 'wcst_O17', kstX)
            call getsrp (xMod, 'aVGN_O17', aVGNX)
            call getsrp (xMod, 'lVGN_O17', lVGNX)
            call getsrp (xMod, 'nVGN_O17', nVGNX)
         else
            call getsrp (xMod, 'wcst_O16', wcstX)
            call getsrp (xMod, 'wcst_O16', kstX)
            call getsrp (xMod, 'aVGN_O16', aVGNX)
            call getsrp (xMod, 'lVGN_O16', lVGNX)
            call getsrp (xMod, 'nVGN_O16', nVGNX)
          end if
      else
      end if
   end if
   do i = istart, nl
      if (pldep - ulimit(i) > 1.e-6) then
         wcst(i) = wcstX
         kst(i) = kstX
         VGA(i) = aVGNX
         VGL(i) = lVGNX
         VGN(i) = nVGNX
         VGR(i) = 0.01
      else
         goto 101
      end if
   end do
   101 call cmdelvar('WCST')
   call cmdelvar('KST')
   call cmdelvar('VGR')
   call cmdelvar('VGA')
   call cmdelvar('VGN')
   call cmdelvar('VGL')
   call putarp (xMod, 'VGR',VGR, Nl)
   call putarp (xMod, 'WCST',WCST, Nl)
```

```
202   call putarp (xMod, 'KST',KST, Nl)
203   call putarp (xMod, 'VGA',VGA, Nl)
204   call putarp (xMod, 'VGN',VGN, Nl)
205   call putarp (xMod, 'VGL',VGL, Nl)
206   CALL SUWCMS (xLogFileUnit,istart,2,WCFCX,FIELD)
207   CALL SUWCMS (xLogFileUnit,istart,2,WCWPX,WILTP)
208   CALL SUWCMS (xLogFileUnit,istart,2,WCADX,AIRDR)
209   CALL SUWCMS (xLogFileUnit,istart,2,WCWUX,WatUp)
210   CALL SUWCMS (xLogFileUnit,istart,2,WCWOX,WatOx)
211   do i = istart, nl
212      if (ulimit(i) < pldep) then
213         wcwp(i) = wcwpX
214         wcfc(i) = wcfcx
215         wcad(i) = wcadx
216         wcwu(i) = wcwux
217         wcwo(i) = wcwox
218         if (llimit(i) > pldep) then
219            pldep = llimit(i)
220            call cmdelvar ('pldep')
221            call putsrp ('GetNewProfile', 'pldep', pldep)
222            goto 102
223         end if
224      else
225         goto 102
226       end if
227   end do
228   102 call cmdelvar('WCFC')
229   call cmdelvar('WCAD')
230   call cmdelvar('WCWP')
231   call cmdelvar('WCWU')
232   call cmdelvar('WCWO')
233   call putarp (xMod, 'WCFC',WCFC, Nl)
234   call putarp (xMod, 'WCAD',WCAD, Nl)
235   call putarp (xMod, 'WCWP',WCWP, Nl)
236   call putarp (xMod, 'WCWU',WCWU, Nl)
237   call putarp (xMod, 'WCWO',WCWO, Nl)
238   end subroutine PedoTrans_VGN
```

I.13. Subroutine GETNEWVANGENUCHTENPARAMSSAND

```
subroutine GetNewVanGenuchtenParamsSand(npGenuchtenSand, &
   pGenuchtenSand, m50, leem, humus, bovengrond, &
   dichtheid, wcst, KSt, aVGN, lVGN, nVGN)
implicit none
integer :: npGenuchtenSand
real :: pGenuchtenSand(npGenuchtenSand)
real :: KStar, aStar, lStar, nStar, DStar, wcst, Kst, aVGN, lVGN, nVGN
real ::  dichtheid, m50, leem, humus, bovengrond
DStar = pGenuchtenSand(43) + pGenuchtenSand(44) * Humus + &
        pGenuchtenSand(45) * Bovengrond + &
        pGenuchtenSand(46) * M50 + pGenuchtenSand(47) * (Leem**2) + &
        pGenuchtenSand(48) / M50 + pGenuchtenSand(49) * alog(M50)
dichtheid = 1. / DStar
wcst  = pGenuchtenSand(1) + pGenuchtenSand(2) * dichtheid + &
        pGenuchtenSand(3) * M50 + pGenuchtenSand(4) * (M50**2) + &
        pGenuchtenSand(5) / Leem + pGenuchtenSand(6) / Humus + &
        pGenuchtenSand(7) / M50 + pGenuchtenSand(8) * alog(Humus) + &
        pGenuchtenSand(9) * alog(M50)
KStar = pGenuchtenSand(10) + pGenuchtenSand(11) * dichtheid + &
        pGenuchtenSand(12) * (Leem**2)+ pGenuchtenSand(13)/ dichtheid &
        + pGenuchtenSand(14)*alog(Leem)+pGenuchtenSand(15)*alog(Humus)
aStar = pGenuchtenSand(16) + pGenuchtenSand(17) * dichtheid + &
        pGenuchtenSand(18) * Bovengrond +  pGenuchtenSand(19) * M50 + &
        pGenuchtenSand(20) / dichtheid + pGenuchtenSand(21) / Leem + &
        pGenuchtenSand(22) * alog(Leem)
lStar = pGenuchtenSand(23) + pGenuchtenSand(24) * Leem + &
        pGenuchtenSand(25) * dichtheid + pGenuchtenSand(26) * M50 +
        pGenuchtenSand(27) * (dichtheid**2) + pGenuchtenSand(28) * &
        (Leem**2) + pGenuchtenSand(29) * (Humus**2) + &
        pGenuchtenSand(30) * (M50**2) + pGenuchtenSand(31)/dichtheid+ &
        pGenuchtenSand(32) / Leem + pGenuchtenSand(33) * alog(Leem) + &
        pGenuchtenSand(34) * alog(M50)
nStar = pGenuchtenSand(35) + pGenuchtenSand(36) * Humus + &
        pGenuchtenSand(37) * dichtheid + pGenuchtenSand(38)*(Leem**2) &
        + pGenuchtenSand(39) / Humus + pGenuchtenSand(40) / M50 + &
        pGenuchtenSand(41) * alog(Humus) + pGenuchtenSand(42) * &
        dichtheid * Leem
KSt   = exp(KStar)
aVGN  = exp(aStar)
nVGN  = 1. + exp(nStar)
lVGN  = 2. * (exp(lStar) - 1.) / (exp(lStar) + 1.)
end subroutine GetNewVanGenuchtenParamsSand
```

I.14. Subroutine GETNEWVANGENUCHTENPARAMSCLAY

```
1   subroutine GetNewVanGenuchtenParamsClay(npGenuchtenClay, &
2      pGenuchtenClay, Lutum, Humus, dichtheid, wcst, KSt, aVGN, lVGN,nVGN)
3   implicit none
4   integer :: npGenuchtenClay
5   real :: pGenuchtenClay(npGenuchtenClay)
6   real :: KStar, aStar, lStar, nStar, DStar, wcst, Kst, aVGN, lVGN, nVGN
7   real :: dichtheid, Lutum, Humus
8   DStar = pGenuchtenClay(27) + pGenuchtenClay(28) * Lutum + &
9      pGenuchtenClay(29) * (Humus**2) + pGenuchtenClay(30) * alog(Humus)
10  dichtheid = 1. / DStar
11  wcst  = pGenuchtenClay(1) + pGenuchtenClay(2) * Lutum + &
12     pGenuchtenClay(3) * (dichtheid**2)+pGenuchtenClay(4)*Dichtheid*Lutum
13  KStar = pGenuchtenClay(5) + pGenuchtenClay(6) * Humus + &
14     pGenuchtenClay(7) * dichtheid + pGenuchtenClay(8) *(dichtheid**2)+ &
15     pGenuchtenClay(9) * (Humus**2) + pGenuchtenClay(10) * Lutum *Humus &
16     + pGenuchtenClay(11) * dichtheid * Humus
17  aStar = pGenuchtenClay(12) + pGenuchtenClay(13) * Humus + &
18     pGenuchtenClay(14) * dichtheid + pGenuchtenClay(15)*(dichtheid**2) &
19     + pGenuchtenClay(16) / Humus + pGenuchtenClay(17) * alog(Humus) + &
20     pGenuchtenClay(18) * dichtheid * Humus
21  lStar = pGenuchtenClay(19) + pGenuchtenClay(20) * Lutum + &
22     pGenuchtenClay(21) * dichtheid * Lutum
23  nStar = pGenuchtenClay(22) + pGenuchtenClay(23) / dichtheid + &
24     pGenuchtenClay(24) * alog(Lutum) + pGenuchtenClay(25) *alog(Humus) &
25     + pGenuchtenClay(26) * Dichtheid * Humus
26  KSt   = exp(KStar)
27  aVGN  = exp(aStar)
28  nVGN  = 1. + exp(nStar)
29  lVGN  = 10. * (exp(lStar) - 1.) / (exp(lStar) + 1.)
30  end subroutine GetNewVanGenuchtenParamsClay
```

I.15. Subroutine LUPT3BPOS

```
subroutine lupt3bPos (xNewTask,xModule,xLogFileUnit)
implicit none
include '..\GetCropData\CropDataSet.inc'
character*(*) xNewTask,xModule
integer xLogFileUnit, LeafClOld, LeafClOldP, i, GroundCoverExt, FintExt
real LeafWtGrowthP, LeafArGrowthPlP, LeafArSum, StemWtGrowthP, Delt
real TuberWtGrowthP, RootDepthChangeP, RootHeightChangeP
real LeafArGrowthP, RConvLossReserve, AboveDryWtGrowthP,
real SourcePotP, SourcePotLUE, SourcePotPMX, RConvLossRedistr
real AboveGlucoseUseP, StemVolGrowthP, RelStemVolGrowthMx, RelGrowth
real leaflosswt, laimx, TMDA,RDD, DAYL, TMDAD, STVol, STVolLFAreaRatio
real StemVol LeafArGrowthPl1, LeafArGrowthPl2,, FINT, FINTTOT
real FINTTOTRED, ECPDFObsvRED, FINTTOTGRN, ECPDFObsvGRN, ECPDFReal
real ECPDFObsv, LUE_oijen, AlphaTmEffLvAge, BetaTmEffLvAge, TMEff, PAR
real LUEPMX, TMEffGrowth, TMEffGrowthPMX, TMEffLvAge, TmeffLvFormation
real AlpTmEffLvFormation, BetTmEffLvFormation, PARTrans, LeafAge, PARAV
real AlphaTmEffLvArF,BetaTmEffLvArf, alphaRelDeadStem, gammaRelDeadStem
real StressIndexRtWt, StressIndexStWt, StressIndexTuWt, StressIndexLvAr
real StressIndexRtDp, StressIndexRtHt, StressIndexTmSu, StressIndexStAg
real tressIndexRoAg, StressIndexTuAg, StressIndexLEff, StressIndexLUE
real StressIndexPAMAX, StressIndexECPDF, StressIndexPhot, ReserveDMTot
real RedistrDMTot, ConvReserveDM, FrReserveUseP, FrRedistrUseP
real GVILeaves, GVIStems, GVITubers, GVIReserves, FreeDMLeafLive
real FreeDMStemLive, FreeDMTuberLive, ReserveLeafLive, ReserveStemLive
real ReserveTuberLive, RedistrLeafDead,RedistrStemDead,RedistrRootDead
real RedistrTuberDead, kdestableaf, kdestabstem, kdestabtuber
real frreserveroot, frreserveleaf, frreservestem, frreservetuber
real FrDyingLeaves, TotDeadLeafRate, StrucDMTuberLive, MaxRelDeadStem
real PAMAX, EFF, PMAX, gammaPAMAX, StemWtX, StemDeadWtX, FintStem
real ConvPhot
character*30 :: FINTtext, SCANtext, SCANtextRED, SCANtextGRN
real, parameter :: CO2A = 350., O2 = 21.
real    intgrl,lint2,limit, funcecpdf, Phot_Oijen
integer ifindc
character*13 OldTask, EmptyString
logical :: DoNitroStress
character*30 :: namerates(1)
save
data namerates / 'LeafArPl'/
if (xNewTask == 'initialize') then
   call Logmessage (xLogFileUnit,9,xModule,'3.0')
   call getsrp (xModule,'delt',Delt)
   call getsrp (xModule, 'RGRL',RGRL)
   call getsrp (xModule, 'alphaT' ,alphaT)
   call getsrp (xModule, 'betaT' ,betaT)
   call getsrp (xModule, 'gammaT' ,gammaT)
   call getsrp (xModule, 'deltaT' ,deltaT)
   call getsrp (xModule, 'AlphaTmEffLvArf' , AlphaTmEffLvArf)
   call getsrp (xModule, 'BetaTmEffLvArf'  , BetaTmEffLvArf)
   call getsrp (xModule, 'AlphaTmEffLvAge' , AlphaTmEffLvAge)
   call getsrp (xModule, 'BetaTmEffLvAge'  , BetaTmEffLvAge)
   call getsrp (xModule, 'AlpTmEffLvFormation' ,AlpTmEffLvFormation)
   call getsrp (xModule, 'BetTmEffLvFormation' ,BetTmEffLvFormation)
   call getsrp (xModule, 'alphaLvAgeT' ,alphaLvAgeT)
   call getsrp (xModule, 'betaLvAgeT' ,betaLvAgeT)
   call getsrp (xModule, 'PARFraction',PARFraction)
   call getsrp (xModule, 'LUE',LUE)
   call getsrp (xModule, 'PAMAX',PAMAX)
   call getsrp (xModule, 'gammaPAMAX',gammaPAMAX)
   call getsrp (xModule, 'Eff',Eff)
   call getsrp (xModule, 'RootDepthCropMx', RootDepthCropMx)
   call getsrp (xModule, 'RootDepthGrowthPar', RootDepthGrowthPar)
   call getsrp (xModule, 'TMBase',TMBase)
   call getsrp (xModule, 'LeafAgeMx', LeafAgeMx)
   call getsrp (xModule, 'LeafArGrowthRef', LeafArGrowthRef)
```

```
      call getsrp (xModule, 'MaxRelDeadStem', MaxRelDeadStem)
      call getsrp (xModule, 'alphaRelDeadStem', alphaRelDeadStem)
      call getsrp (xModule, 'gammaRelDeadStem', gammaRelDeadStem)
      call getsrp (xModule, 'ConvReserveDM', ConvReserveDM)
      call getsrp (xModule, 'GVILeaves', GVILeaves)
      call getsrp (xModule, 'GVIStems', GVIStems)
      call getsrp (xModule, 'GVITubers', GVITubers)
      call getsrp (xModule, 'GVIReserves', GVIReserves)
      call getsrp (xModule, 'StVolLfAreaRatio', StVolLfAreaRatio)
      call getsrp (xModule, 'kdestableaf',kdestableaf)
      call getsrp (xModule, 'kdestabstem',kdestabstem)
      call getsrp (xModule, 'kdestabtuber',kdestabtuber)
      call getsc  (xModule, 'FINTtext', FINTtext)
      call getsc  (xModule, 'SCANtext', SCANtext)
      call getsc  (xModule, 'SCANtextGRN', SCANtextGRN)
      call getsc  (xModule, 'SCANtextRED', SCANtextRED)
      call getsi  (xModule, 'GroundCoverExt', GroundCoverExt)
      call getsi  (xModule, 'FintExt', FintExt)
      call getsrp (xModule, 'relLIntleafdead', relLIntleafdead)
      call getsrp (xModule, 'relLIntstemdead', relLIntstemdead)
      call getsrp (xModule, 'ECPDF', ECPDF)
      call getsrp (xModule, 'ECPDFStem', ECPDFStem)
      call getsrp (xModule, 'ECPDFRed', ECPDFRed)
      call getsrp (xModule, 'ECPDFGrn', ECPDFGrn)
      call getsrp (xModule, 'EffPhotStem', EffPhotStem)
      call getslm (xModule, 'DoNitroStress', DoNitroStress, .false.)
      if (not(donitrostress)) then
         call getsrp (xModule, 'frreserveleaf',frreserveleaf)
         call getsrp (xModule, 'frreservestem',frreservestem)
         call getsrp (xModule, 'frreservetuber',frreservetuber)
      end if
      finttot         = 0.
      finttotRED      = 0.
      finttotGRN      = 0.
      Fint            = 0.
      StressIndexLue  = 0.
      StressIndexPAMAX = 0.
      StressIndexPhot = 0.
      StressIndexLvAr = 0.
      StressIndexStAg = 0.
      ReserveDMTot    = 0.
      RedistrDMTot    = 0.
      SourcePotP      = 0.
      FrReserveUseP   = 0.
      FrRedistrUseP   = 0.
      LeafArGrowthP   = 0.
      LeafArGrowthPlP = 0.
      LeafWtGrowthP   = 0.
      TuberWtGrowthP  = 0.
      StemWtGrowthP   = 0.
      StemVolGrowthP  = 0.
      RootDepthChangeP= 0.
      RootHeightChangeP= 0.
      LeafClOldP      = 0
      do i = 1, leafclmxn
         RLeafAgeClP(i) = 0.
      end do
      call putsi  (xModule, 'LeafClOldP', LeafClOldP)
      call astro (xNewTask,'Astro',xLogFileUnit)
   else if (xNewTask == 'do_rates') then
      LeafWtGrowthP    = 0.
      LeafArGrowthP    = 0.
      TuberWtGrowthP   = 0.
      StemWtGrowthP    = 0.
      StemVolGrowthP   = 0.
      RootDepthChangeP = 0.
      RootHeightChangeP = 0.
      LeafArGrowthPlP  = 0.
```

```
      SourcePotP       = 0.
      Fint             = 0.
      Finttot          = 0.
      FreeDMLeafLive   = 0.
      FreeDMStemLive   = 0.
      FreeDMTuberLive  = 0.
      ReserveLeafLive  = 0.
      ReserveStemLive  = 0.
      ReserveTuberLive = 0.
      RedistrLeafDead  = 0.
      RedistrStemDead  = 0.
      RedistrTuberDead = 0.
      ReserveDMTot     = 0.
      RedistrDMTot     = 0.
      FrReserveUseP    = 0.
      FrRedistrUseP    = 0.
      call astro (xNewTask,'Astro',xLogFileUnit)
      call getsrt (xModule, 'dayl'  , dayl)
      if (leafclcnt > 0) then
         call getsrt (xModule, 'StressIndexLUE'  , StressIndexLUE )
         call getsrt (xModule, 'StressIndexPAMAX'  , StressIndexPAMAX )
         call getsrt (xModule, 'StressIndexPhot'  , StressIndexPhot )
         call getsrt (xModule, 'StressIndexECPDF', StressIndexECPDF)
         call getsrt (xModule, 'StressIndexLEff' , StressIndexLEff)
         call getsrt (xModule, 'StressIndexLvAr' , StressIndexLvAr)
         call getsrt (xModule, 'StressIndexStAg' , StressIndexStAg)
         call getsrt (xModule, 'StressIndexRoAg' , StressIndexRoAg)
         call getsrt (xModule, 'StressIndexTuAg' , StressIndexTuAg)
         call getsrt (xModule, 'StressIndexStWt' , StressIndexStWt)
         call getsrt (xModule, 'StressIndexRtWt' , StressIndexRtWt)
         call getsrt (xModule, 'StressIndexTuWt' , StressIndexTuWt)
         call getsrt (xModule, 'StressIndexRtDp' , StressIndexRtDp)
         call getsrt (xModule, 'StressIndexRtHt' , StressIndexRtHt)
         call getsrt (xModule, 'StressIndexTmSu' , StressIndexTmSu)
         call getsi (xModule, 'LeafClOLd',  LeafClOld)
         call getsrt (xModule, 'LeafWt'  , LeafWt )
         call getsrt (xModule, 'LeafDeadWt'  , LeafDeadWt)
         call getsrt (xModule, 'LeafLossWt'  , LeafLossWt)
         call getsrt (xModule, 'StemWt'  , StemWt )
         call getsrt (xModule, 'StemVol' , StemVol )
         call getsrt (xModule, 'StemDeadWt', StemDeadWt)
         call getsrt (xModule, 'LaiTot'  , LaiTot)
         call getsrt (xModule, 'LaiMx'  , LaiMx)
         call getsrt (xModule, 'LaiLive', LaiLive)
         call getsrt (xModule, 'LaiDead', LaiDead)
         call getsrt (xModule, 'LaiEff', LaiEff)
         call getsrt (xModule, 'LeafArPl', LeafArPl)
         if (donitrostress) then
            call getsrtm(xModule, 'FreeDMStemLive',   FreeDMStemLive, 0.)
            call getsrtm(xModule, 'FreeDMTuberLive',  FreeDMTuberLive, 0.)
            StrucDMTuberLive = max(0.,TuberWt - FreeDMTuberLive)
            call putsrt(xModule, 'StrucDMTuberLive', StrucDMTuberLive)
         else
            FreeDMStemLive   = frreservestem * StemWt
            FreeDMTuberLive  = frreserveTuber * TuberWt
            StrucDMTuberLive = max(0.,TuberWt - FreeDMTuberLive)
            do i = 1, leafclcnt
               FreeDMleafCl(i) = frreserveleaf * LeafWtLiveCl(i)
            end do
            call cmdelvar('FreeDMStemLive')
            call cmdelvar('FreeDMTuberLive')
            call putsrt(xModule, 'FreeDMStemLive', FreeDMStemLive)
            call putsrt(xModule, 'FreeDMTuberLive', FreeDMTuberLive)
            call putsrt(xModule, 'StrucDMTuberLive', StrucDMTuberLive)
         end if
         call getsrt (xModule,'TMDA',TMDA)
         call getsrt (xModule,'TMDAD',TMDAD)
         TMEff = MAX (0., (TMDA-TmBase)*(1.-exp(max(-32., min(32., &
```

```
202             AlphaTMEffLvArf * (TMDA - BetaTmEffLvArf))))))
203          TMEffGrowth = max(0., (1./(1.+exp(max(-30.,min(30.,-alphaT * &
204                 (TMDA-betaT)))))) * (1./(1.+exp(max(-30.,min(30., &
205                 -gammaT*(deltaT-TMDA)))))))
206          TMEffGrowthPMX = max(0., (1./(1.+exp(max(-30.,min(30., &
207                 -alphaT*(TMDAD-betaT)))))) * (1./(1.+exp(max( &
208                 -30.,min(30.,-gammaT*(deltaT-TMDAD)))))))
209          if (StemWt > 0. ) then
210             StemWtX = StemWt**(2./3.)
211          else
212             StemWtX = 0.
213          end if
214          if (StemDeadWt > 0.) then
215             StemDeadWtX = StemDeadWt**(2./3.)
216          else
217             StemDeadWtX = 0.
218          end if
219          FINT       = 1.-exp (max(-30.,min(30.,-ECPDF * (LAILive + &
220                       ecpdfstem * StemWtX))))
221          FINTTOT    = 1.-exp (max(-30.,min(30.,-ECPDF * (LaiLive + &
222                       relLIntleafdead*LaiDead + ecpdfstem * (StemWtX + &
223                       relLIntstemdead*StemDeadWtX)))))
224          FINTTOTRED = 1.-exp (max(-30.,min(30.,-ECPDFRED * (LaiLive + &
225                       relLIntleafdead*LaiDead + ecpdfstem * (StemWtX + &
226                       relLIntstemdead*StemDeadWtX)))))
227          FINTTOTGRN = 1.-exp (max(-30.,min(30.,-ECPDFGRN * (LaiLive + &
228                       relLIntleafdead*LaiDead + ecpdfstem * (StemWtX + &
229                       relLIntstemdead*StemDeadWtX)))))
230          call getsrt (xModule,'RDD' ,RDD)
231          PAR        = PARFraction * RDD
232          FintStem = max(0., min (1., FINT - (1.-exp (max(-30.,min(30., &
233                     -ECPDF * LaiLive))))))
234          SourcePotLUE  = PAR * ((FINT-FintStem)+ FintStem * EffPhotStem)*&
235                          LUE * StressIndexLue * TMEffGrowth*10./1.e6
236          PMAX = PAMAX * StressIndexPAMAX
237          PARAV = (PAR * 1.0E-06) / (DAYL/24.)
238          LUEPMX = TMEffGrowthPMX * (gammaPAMAX * EFF * PMAX)/(EFF *ECPDF &
239                   * PARAV + PMAX )
240          SourcePotPMX = PAR * ((FINT-FintStem)+ FintStem * EffPhotStem)* &
241                   LUEPMX  * 10. * 1.e-6
242          ConvPhot = 1.0
243          LUE_oijen = Phot_Oijen(CO2A, O2, ConvPhot, PARAV, TMDAD, &
244                      StressIndexPhot, ECPDF)
245          SourcePotP = PAR * ((FINT-FintStem)+ FintStem * EffPhotStem) * &
246                       LUE_oijen  * 10. * 1.e-6
247          RedistrLeafDead = 0.
248          ReserveLeafLive = 0.
249          LeafArSum = 0.
250          totdeadleafrate = 0.
251          TmEffLvAge = alphaTMEffLvAge * exp(max(-30., min(30., &
252                       betaTMEffLvAge * TMDA)))
253          do i=LeafClCnt, 1, -1
254             StressIndexLvAgClP(i) = StressIndexLvAgCl(i)
255             if (i < LeafClCnt) LeafArSum = LeafArSum + LeafArLiveCL(i+1)+&
256                LeafArDeadCl(i+1)
257             PARTrans = PAR * exp (-ECPDF * LeafArSum / 10000.)
258             RLeafAgeClP(i) =  max(0., StressIndexLvAgClP(i)*(alphalvageT &
259                - betalvageT * PARTrans / 15.e6 ) * TmEffLvAge * delt)
260             if ((LeafAgeCl(i) > LeafAgeMx) .and. (LeafAliveCl(i)) ) then
261                LeafAliveCl(i) = .false.
262                LeafClOldP   = max(leafClold, i)
263                RedistrLeafDead = RedistrLeafDead + FreeDMleafCl(i)
264                TotDeadLeafRate = TotDeadLeafRate + LeafWtLiveCl(i)
265             elseif (LeafAliveCl(i)) then
266                ReserveLeafLive    = ReserveLeafLive + FreeDMleafCl(i) * &
267                   kdestableaf
268             end if
269          end do
```

```
      FrDyingLeaves    = TotDeadLeafRate / max(1.e-6, LeafWt)
      if (not(DoNitroStress)) then
         StressIndexStAg = 0.
         if (LaiLive < LaiMx) then
            if (leafwt / max(1.e-10,leafwt + leafdeadwt + leaflosswt) &
               <= 0.) then
               StressIndexStAg = min(1., max(0., MaxRelDeadStem ))
            elseif(leafwt / max(1.e-10,leafwt + leafdeadwt + &
               leaflosswt) <= gammareldeadstem) then
               StressIndexStAg = min(1., max(0., MaxRelDeadStem * (1.-&
                   min(1.,leafwt / max(1.e-10,leafwt + leafdeadwt + &
                  leaflosswt) / max(1.e-10, gammareldeadstem))) &
                  **alphaRelDeadStem))
            else
               StressIndexStAg = 0.
            end if
         endif
      end if
      ReserveStemLive  = (1. - StressIndexStAg) * FreeDMStemLive * &
         kdestabstem
      ReserveTuberLive = (1. - StressIndexTuAg) * FreeDMTuberLive * &
         kdestabtuber
      RedistrStemDead  = StressIndexStAg * FreeDMStemLive
      RedistrTuberDead = StressIndexTuAg * FreeDMTuberLive
      RedistrDMTot  = RedistrLeafDead+RedistrStemDead+RedistrTuberDead
      ReserveDMTot  = ReserveLeafLive+ReserveStemLive+ReserveTuberLive
      TmeffLvFormation =  exp(max(-30., min(30., -AlpTmEffLvFormation*&
         (TMDA-BetTmEffLvFormation)**2)))
      LeafArGrowthPl1 = LeafArPl * (exp (max(-30., min(30., RGRL * &
          TmEff * Delt)))-1.)
      LeafArGrowthPl2 = LeafArGrowthRef * TmeffLvFormation
      call GetLeafStemRatio
      call GetSLA(SLA)
      call GetSTVol(STVol)
      LeafArGrowthPlP = StressIndexLvAr * min(LeafArGrowthPl1, &
                        LeafArGrowthPl2)
      if (LeafArGrowthPlP < 1.e-6) LeafArGrowthPlP = 0.
      call getsrp (xModule, 'NPL', NPL)
      LeafArGrowthP     = LeafArGrowthPlP * NPL
      LeafWtGrowthP     = (LeafArGrowthP / 10000.) / max(1.e-6, SLA)
      if (LaiLive < LaiMx) then
         if (leafwt / max(1.e-10,leafwt + leafdeadwt + leaflosswt) <= &
            gammareldeadstem) then
            StemVolGrowthP    = 0.
         else
            StemVolGrowthP = StemVol * RelStemVolGrowthMx * (leafwt / &
               max(1.e-10,leafwt + leafdeadwt + leaflosswt) / &
               max(1.e-10, gammareldeadstem))
         end if
      else
         StemVolGrowthP   = StVolLfAreaRatio * (LeafArGrowthP / 10000.)
         RelStemVolGrowthMx = max(0., min(1., StemVolGrowthP / &
            max(1.e-10, StemVol)))
      endif
      StemWtGrowthP     = StemVolGrowthP * StressIndexStWt / StVol
      AboveDryWtGrowthP = LeafWtGrowthP + StemWtGrowthP
      AboveGlucoseUseP = LeafWtGrowthP*GVILeaves+StemWtGrowthP*GVIStems
      FrReserveUseP    = 0.
      FrRedistrUseP    = 0.
      RConvLossReserve = 0.
      RConvLossRedistr = 0.
      if (SourcePotP < AboveGlucoseUseP) then
         FrRedistrUseP = min(1.,max(0.,(AboveGlucoseUseP-SourcePotP) /&
            max(1.e-10, RedistrDMTot * GVIReserves * ConvReserveDM)))
         if (SourcePotP+FrRedistrUseP*ConvReserveDM*RedistrDMTot* &
             GVIReserves < AboveGlucoseUseP) then
            FrReserveUseP    = min(1., max(0., (AboveGlucoseUseP - &
               SourcePotP - FrRedistrUseP * ConvReserveDM * &
```

```
                RedistrDMTot * GVIReserves) / max(1.e-10, ReserveDMTot &
              * GVIReserves * ConvReserveDM)))
          end if
          RelGrowth = max(0.,(SourcePotP+GVIReserves*ConvReserveDM*&
             ((FrRedistrUseP*RedistrDMTot)+(FrReserveUseP*ReserveDMTot)&
             ) ) / max(1.e-6, AboveGlucoseUseP))
          LeafArGrowthPlP   = LeafArGrowthPlP * RelGrowth
          LeafArGrowthP     = LeafArGrowthPlP * NPL
          LeafWtGrowthP     = (LeafArGrowthP / 10000.) / max(1.e-6, SLA)
          StemVolGrowthP    = RelGrowth * StemVolGrowthP
          StemWtGrowthP     = RelGrowth * StemWtGrowthP
          AboveDryWtGrowthP = LeafWtGrowthP + StemWtGrowthP
          AboveGlucoseUseP  = LeafWtGrowthP * GVILeaves + StemWtGrowthP&
                              * GVIStems
       end if
       TuberWtGrowthP   = max(0., (SourcePotP  + GVIReserves  * ( &
         (FrRedistrUseP * RedistrDMTot ) + (FrReserveUseP * &
          ReserveDMTot) )  - AboveGlucoseUseP ) / GVITubers )
       RootDepthChangeP  = min (RootDepthGrowthPar, &
          max (RootDepthCropMx - RootDepth, 0.)) * StressIndexRtDp
       RootHeightChangeP = -RootDepthGrowthPar * StressIndexRtHt
       call putsi  (xModule, 'LeafClOldP', LeafClOldP)
       call putsrt (xModule, 'TMEff'     , TMEff)
       call putsrt (xModule, 'ECPDFReal' , ECPDFReal)
       call putsrt (xModule, 'ECPDFObsv' , ECPDFObsv)
       call cmdelvar('SLA')
       call putsrp (xModule, 'SLA'      , SLA)
       call putsrt (xModule, 'StressIndexStAgP' , StressIndexStAg)
       call putsrt (xModule, 'StressIndexRoAgP' , StressIndexRoAg)
       call putsrt (xModule, 'StressIndexTuAgP' , StressIndexTuAg)
    end if
    call putsrt (xModule, 'SourcePotP'       , SourcePotP)
    call putsrt (xModule, 'FrReserveUseP'    , FrReserveUseP)
    call putsrt (xModule, 'FrRedistrUseP'    , FrRedistrUseP)
    call putsrt (xModule, 'LeafArGrowthP'    , LeafArGrowthP)
    call putsrt (xModule, 'LeafArGrowthPlP'  , LeafArGrowthPlP)
    call putsrt (xModule, 'LeafWtGrowthP'    , LeafWtGrowthP)
    call putsrt (xModule, 'TuberWtGrowthP'   , TuberWtGrowthP)
    call putsrt (xModule, 'StemWtGrowthP'    , StemWtGrowthP)
    call putsrt (xModule, 'StemVolGrowthP'   , StemVolGrowthP)
    call putsrt (xModule, 'RootDepthChangeP' , RootDepthChangeP)
    call putsrt (xModule, 'RootHeightChangeP' , RootHeightChangeP)
    call putsrt (xModule, 'FINT'             , FINT)
    call putsrt (xModule, 'FINTTOT'          , FINTTOT)
    call putsrt (xModule, 'FINTTOTRED'       , FINTTOTRED)
    call putsrt (xModule, 'FINTTOTGRN'       , FINTTOTGRN)
    call writesrtatdebug(xlogfileunit, xModule, "rates", 1, namerates)
 else if (xNewTask == 'output') then
    call outdat (2,0,'fint',              fint)
    call outdat (2,0,'finttot',           finttot)
 else if (xNewTask == 'do_states') then
    call astro (xNewTask,'Astro',xLogFileUnit)
 else if (xNewTask == 'export_states') then
    call putsrt (xModule, 'FINT'     , FINT)
    call putsrt (xModule, 'FINTTOT'     , FINTTOT)
 end if
 return
 end subroutine lupt3bPos
```

I.16. Subroutine GETSLA

```
subroutine GetSLA(SLA)
implicit none
include '..\GetCropData\CropDataSet.inc'
real DMNStabLeafMax, NSolConc, NStabConcLeafLive
real NstrucLeafMin, NStabLeafMax, DMNStabLeafMin, DMNStrucLeaf
real DMNStabLeaf, alphastabLeaf, betastabLeaf
character*30, parameter :: xModule = 'GetSLA'
logical :: DoNitroStress
call getslm (xModule, 'DoNitroStress', DoNitroStress, .false.)
if (not(DoNitroStress)) then
   call getsrp (xModule, 'SLA' ,SLA)
else
   call getsrp (xModule, 'NSolConcI',     NSolConcI)
   call getsrp (xModule, 'NstrucLeafMin',  NstrucLeafMin)
   call getsrp (xModule, 'DMNstrucLeaf',   DMNstrucLeaf)
   call getsrp (xModule, 'NstabLeafMax',   NstabLeafMax)
   call getsrp (xModule, 'DMNstabLeafMin', DMNstabLeafMin)
   call getsrp (xModule, 'DMNstabLeafMax', DMNstabLeafMax)
   call getsrp (xModule, 'alphastabLeaf',  alphastabLeaf)
   call getsrp (xModule, 'betastabLeaf',   betastabLeaf)
   call getsrtm (xModule, 'NSolConc',      NSolConc, NSolConcI)
   DMNStabLeaf = DMNStabLeafMin + (DMNStabLeafMax - DMNStabLeafMin) / &
      (1. + exp(max(-30., min(30., alphaStabLeaf * (NSolConc - &
       betastableaf)))))
   SLA = 1./max(1.e-10, NstrucLeafMin * DMNStrucLeaf + NStabLeafMax * &
      DMNStabLeaf)
end if
end subroutine GetSLA
```

I.17. Subroutine GETSTVOL

```
subroutine GetSTVol(STVol)
implicit none
include '..\GetCropData\CropDataSet.inc'
real STVol, alphaStabStem, LeafWtGrowth, LAIGrowth, StVolLfAreaRatio
real NSolConc, NstrucStemMin, NStabStemMax, DMNStabStemMin
real DMNStabStemMax, DMNStabStem, DMNStrucStem, betastabStem
real NuptR, sourcepotp, NStabConcStemVol
character*30, parameter :: xModule = 'GetSTVol'
logical :: DoNitroStress
call getslm (xModule, 'DoNitroStress', DoNitroStress, .false.)
if (not(DoNitroStress)) then
   call getsrp (xModule, 'LeafStemRatio' ,LeafStemRatio)
   call getsrp (xModule, 'StVolLfAreaRatio' ,StVolLfAreaRatio)
   call GetSLA(SLA)
   if (SLA > 0.) then
      STVol = SLA * StVolLfAreaRatio / max(1.e-6, LeafStemRatio)
   else
      STVol = 0.
   end if
else
   call getsrp (xModule, 'NstrucStemMin',  NstrucStemMin)
   call getsrp (xModule, 'DMNstrucStem',   DMNstrucStem)
   call getsrp (xModule, 'NstabStemMax',   NstabStemMax)
   call getsrp (xModule, 'DMNstabStemMin', DMNstabStemMin)
   call getsrp (xModule, 'DMNstabStemMax', DMNstabStemMax)
   call getsrp (xModule, 'alphastabStem',  alphastabStem)
   call getsrp (xModule, 'betastabStem',   betastabStem)
   call getsrtm (xModule, 'NSolConc',       NSolConc, NSolConcI)
   DMNStabStem = DMNStabStemMin + (DMNStabStemMax - DMNStabStemMin) / &
      (1. + exp(max(-30., min(30., alphaStabStem * (NSolConc - &
       betastabstem)))))
   STVol = 1./max(1.e-10, NstrucStemMin * DMNStrucStem + NStabStemMax &
      * DMNStabStem)
end if
end subroutine GetSTVol
```

I.18. Real function PHOT_OIJEN

```
Real function Phot_Oijen(CO2A,O2,Y,PARAV, PHOTMP, StressIndexPhot, KDF)
! Rodriguez, D., M. van Oijen & A.H.M.C. Schapendonk, 1999. LINGRA-CC:&
! a sink-source model to simulate the impact of climate change and &
! management on grassland productivity. New Phytologist 144: 359-368.
implicit none
real, parameter :: JMUMOL = 4.56
real, parameter :: KC25=138., KMC25=460.,  KMO25=33.,  KOKC=0.21
real, parameter :: EAVCMX=68000.,  EAKMC=65800., EAKMO=1400.
real :: Y, PARAV, PHOTMP, PMAXFC, EFFFC, CO2I, VCMAX, KMC, KMO, GAMMAX
real :: StressIndexPhot, CO2A, O2, KDF, MaxRubiscoEff
call getsrp('Phot_Oijen', 'MaxRubiscoEff', MaxRubiscoEff)
CO2I   = 0.7 * CO2A
VCMAX  = MaxRubiscoEff * StressIndexPhot * EXP((1./298. - &
         1./(PHOTMP+273.))*EAVCMX/8.314)
KMC    = KMC25 * EXP((1./298.-1./(PHOTMP+273.)) * EAKMC/8.314)
KMO    = KMO25 * EXP((1./298.-1./(PHOTMP+273.)) * EAKMO/8.314)
GAMMAX = 0.5 * KOKC * KMC * O2 / KMO
PMAXFC = VCMAX * (CO2I-GAMMAX) / (CO2I + KMC * (1.+O2/KMO))
EFFFC = 44. * JMUMOL/2.1 * (CO2I-GAMMAX) / (4.5 * CO2I + 10.5 * GAMMAX)
Phot_Oijen = Y * EFFFC * PMAXFC / (EFFFC*KDF*PARAV + PMAXFC)
end function Phot_Oijen
```

I.19. Subroutine GETLEAFSTEMRATIO

```
subroutine GetLeafStemRatio
implicit none
real LeafStemRatioMax, LeafStemRatioMin, LeafStemRatio, LeafWt, StemWt
real alphaLeafStemRatio, betaLeafStemRatio LeafDeadWt, LeafLossWt
real StemDeadWt, StemLossWt, TotAboveWt
character*30, parameter :: xModule = 'GetLeafStemRatio'
call getsrp (xModule, 'LeafStemRatioMax' ,LeafStemRatioMax)
call getsrp (xModule, 'LeafStemRatioMin' ,LeafStemRatioMin)
call getsrp (xModule, 'alphaLeafStemRatio' ,alphaLeafStemRatio)
call getsrp (xModule, 'betaLeafStemRatio'  ,betaLeafStemRatio)
call getsrtm(xModule, 'LeafWt'    , LeafWt,0.)
call getsrtm(xModule, 'LeafDeadWt', LeafDeadWt,0.)
call getsrtm(xModule, 'LeafLossWt', LeafLossWt,0.)
call getsrtm(xModule, 'StemWt'    , StemWt,0.)
call getsrtm(xModule, 'StemDeadWt', StemDeadWt,0.)
call getsrtm(xModule, 'StemLossWt', StemLossWt,0.)
TotAboveWt =LeafWt+LeafDeadWt+LeafLossWt+StemWt+StemDeadWt+StemLossWt
LeafStemRatio = (LeafStemRatioMax-LeafStemRatioMin) / (1. + exp (min &
   (30., max(-30., -alphaLeafStemRatio * (betaLeafStemRatio - &
   TotAboveWt))))) + LeafStemRatioMin
call cmdelvar ('leafstemratio')
call putsrp ('master','LeafStemRatio', LeafStemRatio)
end subroutine GetLeafStemRatio
```

I.20. Subroutine LUPT3BREA

```
subroutine lupt3bRea (xNewTask,xMod,xLogFileUnit)
implicit none
include '..\GetCropData\CropDataSet.inc'
character*(*) :: xNewTask,xMod
integer xLogFileUnit,iyear, idoy, datecmp, datecmp2, datecmp3, dtfsecmp
integer :: LeafClOld, LeafClOldNew, i,tmpi1,RootLayerNo

real fintim, delt, DayLaiMx, LAIMx, LeafPresentWt, StemPresentWt
real AbovePresentWt, StemVol, StVol, StVolLfAreaRatio, LAILoss
real LeafArGrowthPl, LeafArGrowthPlP, LeafArDead, LeafArGrowthP
real LeafArLoss, StemWtGrowth, StemWtGrowthP, TuberWtGrowth
real TuberWtGrowthP, StemVolGrowth, StemVolGrowthP, RelGrowth
real RootDepthChange, RootDepthChangeP, RootHeightChange
real RootHeightChangeP, TotWtGrowth, LeafArGrowth, SLARTot
real LeafWtClLiveTot, LeafWtClDeadTot,LeafWtClLossTot, LaiLeafKill
real RShufDMLeafLive, RShufDMStemLive, RShufDMTuberLive, RShufDMTot
real RShufDMLeafDead, RShufDMStemDead, RShufDMTuberDead
real, dimension(LeafClMxn) :: LeafAgeClI,RLeafWtLiveCl,RLeafWtDeadCl, &
      RLeafWtLossCl, RLeafArLiveCl, RLeafArDeadCl, RLeafArLossCl
real LeafLossWt, StemDeadWtI, StemLossWt,TuberDeadWtI, TuberLossWt
real GVILeaves, GVIStems, GVITubers, GVIReserves, AboveGlucoseUse
real TMDA,RDD, LeafArGrowthPl1,LeafArGrowthPl2m LeafWtGrowth
real LeafWtGrowthP, LeafAge, FINT, FINTTOT, TMEff, TMEffLvAge, PAR
real TParInt, SourcePot, SourcePotP, SourcePotCum, AboveDryWtGrowth
real AboveDryWtGrowthP, HarvestIndex, TotalDryWt,TotalDryWt_Ro
real AboveDryWt,TotalAboveDryWt, LiveDryWt, NonTuberWt, TotalDryWtI
real ReserveDMTot, ReserveDMTotCum, RedistrDMTot, RedistrDMTotCum
real ConvReserveDM, ConvLossReserve, RConvLossReserve, ConvLossRedistr
real RConvLossRedistr, FrReserveUseP, FrReserveUse, FrRedistrUseP
real FrRedistrUse, FreeDMLeafLive, FreeDMStemLive, FreeDMTuberLive
real StrucDMTuberLive, TuberRespiration, TuberRespirationCum
real RelRespTuber, ReserveLeafLive, ReserveStemLive, ReserveTuberLive
real RedistrLeafDead, RedistrStemDead, RedistrTuberDead, RLeafLiveAr
real RLeafDeadAr, RLeafLossAr, RLeafLiveWt, RLeafDeadWt,  RLeafLossWt
real RStemLiveWt, RStemDeadWt, RStemLossWt, RTuberLiveWt, RTuberDeadWt
real RTuberLossWt, DMIncoming, DMAccounted, DMBalance, alphatuberfresh
real betatuberfresh, OWGcalc, alphaOWG, betaOWG, UBGcalc, alphaUBG
real betaUBG, UBGCalc0(366), StarchCalc, alphaStarch, betaStarch
real StressIndexLvWt, StressIndexRtWt, StressIndexStWt, StressIndexTuWt
real StressIndexLvAr, StressIndexRtDp, StressIndexRtHt, StressIndexTmSu
real StressIndexStAg, StressIndexRoAg, StressIndexTuAg, StressIndexTuGr
realStressIndexStAgP, StressIndexRoAgP, StressIndexTuAgP, kdestabtuber
real StressIndexLEff, StressIndexLUE, kdestableaf, kdestabstem
real intgrl
character*13 OldTask
character*30 :: namestates(1)
save
data namestates /'leafarpl'/
if (xNewTask == 'initialize') then
   call Logmessage (xLogFileUnit,9,xMod,'3.0')
   call getsrp  (xMod,'delt'         ,Delt)
   call getsl   (xMod, 'DoEmergence',   DoEmergence)
   call getsrp (xMod, 'Fintim',          Fintim)
   call getsi  (xMod, 'iyear',           iyear)
   call getsim (xMod, 'CropHarvestYear', CropHarvestYear, 0)
   call getsim (xMod, 'CropHarvestDay',  CropHarvestDay,  0)
   call getsim (xMod, 'LaiLeafKillYear', LaiLeafKillYear, 0)
   call getsim (xMod, 'LaiLeafKillDay', LaiLeafKillDay,   0)
   if (CropHarvestYear <= 0) then
      CropHarvestYear =iyear
      call cmdelvar('CropHarvestYear')
      call putsi (xMod, 'CropHarvestYear', CropHarvestYear)
   end if
   if (CropHarvestDay <= 0) then
      CropHarvestDay = nint(Fintim)
```

```
      call cmdelvar ('CropHarvestDay')
      call putsi (xMod, 'CropHarvestDay', CropHarvestDay)
   end if
   if (LaiLeafKillYear <= 0) then
       LaiLeafKillYear = CropHarvestYear
       call cmdelvar ('LaiLeafKillYear')
       call putsi (xMod, 'LaiLeafKillYear', LaiLeafKillYear)
   end if
   if (LaiLeafKillDay <= 0) then
       LaiLeafKillDay = CropHarvestDay
       call cmdelvar ('LaiLeafKillDay')
       call putsi (xMod, 'LaiLeafKillDay', LaiLeafKillDay)
   end if
   call getsi   (xMod,'LeafClCntI'   ,LeafClCntI)
   call getsrp  (xMod,'LeafArPlI'    ,LeafArPlI)
   call getsrp  (xMod,'RootDepthI'   ,RootDepthI)
   call getsrpm(xMod, 'RootHeightI', RootHeightI, 0.)
   call getsrpm (xMod,'TMSumI'       ,TMSumI, 0.)
   call getsrpm (xMod,'LeafDeadWtI'  ,LeafDeadWtI, 0.)
   call getsrpm (xMod,'StemDeadWtI'  ,StemDeadWtI, 0.)
   call getsrpm (xMod,'TuberDeadWtI'  ,TuberDeadWtI, 0.)
   call getsrpm (xMod,'StemWtI'      ,StemWtI, 0.)
   call getsrpm (xMod,'TuberWtI'     ,TuberWtI, 0.)
   call getsrp (xMod, 'RelLossDeadLeaves',RelLossDeadLeaves)
   call getsrp (xMod, 'RelLossDeadStems',RelLossDeadStems)
   call getsrp (xMod, 'RelLossDeadTubers',RelLossDeadTubers)
   RelLossDeadLeaves  = min(1., max(0., RelLossDeadLeaves))
   RelLossDeadStems   = min(1., max(0., RelLossDeadStems))
   RelLossDeadTubers  = min(1., max(0., RelLossDeadTubers))
   call getsrp (xMod, 'PARFraction',PARFraction)
   call getsrp (xMod, 'StVolLfAreaRatio', StVolLfAreaRatio)
   call getsrp (xMod, 'ConvReserveDM', ConvReserveDM)
   ConvReserveDM = max(0., min(1., ConvReserveDM))
   call getsrp (xMod, 'GVILeaves', GVILeaves)
   call getsrp (xMod, 'GVIStems', GVIStems)
   call getsrp (xMod, 'GVITubers', GVITubers)
   call getsrp (xMod, 'GVIReserves', GVIReserves)
   call getsrp(xMod, 'kdestableaf',kdestableaf)
   call getsrp(xMod, 'kdestabstem',kdestabstem)
   call getsrp(xMod, 'kdestabtuber',kdestabtuber)
   call getsrp(xMod, 'alphatuberfresh',alphatuberfresh)
   call getsrp(xMod, 'betatuberfresh',betatuberfresh)
   call getsrp(xMod, 'alphaOWG',alphaOWG)
   call getsrp(xMod, 'betaOWG',betaOWG)
   call getsrp(xMod, 'alphaUBG',alphaUBG)
   call getsrp(xMod, 'betaUBG',betaUBG)
   call getsrp(xMod, 'alphaStarch',alphaStarch)
   call getsrp(xMod, 'betaStarch',betaStarch)
   call getsrp(xMod, 'RelRespTuber', RelRespTuber)
   call getsi (xMod,'idoy',idoy)
   laiTot          = 0.
   LaiEff          = 0.
   laiLive         = 0.
   laimx           = 0.
   daylaimx        = 0.
   LaiDead         = 0.
   LaiLoss         = 0.
   TuberFreshWt    = 0.
   OWGcalc         = 0.
   UBGcalc         = 0.
   Starchcalc      = 0.
   RootDepth       = 0.
   RootHeight      = 0.
   LeafWt          = 0.
   LeafDeadWt      = 0.
   LeafPresentWt   = 0.
   StemDeadWt      = 0.
   StemPresentWt   = 0.
```

```
      TuberDeadWt     = 0.
      LeafLossWt      = 0.
      StemLossWt      = 0.
      TuberLossWt     = 0.
      RLeafLossWt     = 0.
      RStemLossWt     = 0.
      RTuberLossWt    = 0.
      RLeafDeadWt     = 0.
      RStemDeadWt     = 0.
      RTuberDeadWt    = 0.
      StemWt          = 0.
      StemVol         = 0.
      TuberWt         = 0.
      LeafAr          = 0.
      LeafArPl        = 0.
      TMSum           = 0.
      RootDepth       = 0.
      LiveDryWt       = 0.
      TotalDryWt      = 0.
      TotalDryWt_Ro   = 0.
      NonTuberWt      = 0.
      AboveDryWt      = 0.
      AbovePresentWt  = 0.
      TotalAboveDryWt = 0.
      HarvestIndex    = 0.
      LeafClCnt       = 0
      LeafClOld       = 0
      TParInt         = 0.
      TotalDryWtI     = 0.
      SourcePotCum    = 0.
      TuberRespirationCum = 0.
      RedistrDMTotCum = 0.
      ReserveDMTotCum = 0.
      ConvLossReserve = 0.
      ConvLossRedistr = 0.
      SLA             = 0.
      SLARTot         = 0.
      ReserveDMTot    = 0.
      FrReserveUse    = 0.
      FrRedistrUse    = 0.
      RedistrDMTot    = 0.
      RShufDMTot      = 0.
      AboveGlucoseUse = 0.
      SourcePot       = 0.
      LeafArGrowth    = 0.
      FreeDMLeafLive   = 0.
      FreeDMStemLive   = 0.
      FreeDMTuberLive  = 0.
      StrucDMTuberLive  = 0.
      ReserveLeafLive   = 0.
      ReserveStemLive   = 0.
      ReserveTuberLive  = 0.
      RedistrLeafDead   = 0.
      RedistrStemDead   = 0.
      RedistrTuberDead  = 0.
      RLeafLiveAr  = 0.
      RLeafDeadAr  = 0.
      RLeafLossAr  = 0.
      RLeafLiveWt  = 0.
      RLeafDeadWt  = 0.
      RLeafLossWt  = 0.
      RStemLiveWt  = 0.
      RStemDeadWt  = 0.
      RStemLossWt  = 0.
      RTuberLiveWt = 0.
      RTuberDeadWt = 0.
      RTuberLossWt = 0.
      TotWtGrowth  = 0.
```

```
      LaiLeafKill  = 0.
      TuberRespiration = 0.
      do i = 1, leafclmxn
         RLeafAgeCl(i) = 0.
         LeafAgeCl(i) = 0.
      end do
      call putsrt (xMod, 'LaiEff'      , LaiEff)
      call putsrt (xMod, 'Laitot'      , Laitot)
      call putsrt (xMod, 'Laidead'     , Laidead)
      call putsrt (xMod, 'laiLive'     , laiLive)
      call putsrt (xMod, 'LaiLeafKill' , LaiLeafKill)
      call putsrt (xMod, 'TuberFreshWt', TuberFreshWt)
      call putsrt (xMod, 'OWGCalc'     , OWGCalc)
      call putsrt (xMod, 'UBGCalc'     , UBGCalc)
      call putsrt (xMod, 'RootDepth'   , RootDepth)
      call putsrt (xMod, 'RootHeight'  , RootHeight)
      call putsrt (xMod, 'LeafWt'      , LeafWt)
      call putsrt (xMod, 'LeafDeadWt'  , LeafDeadWt)
      call putsrt (xMod, 'StemWt'      , StemWt)
      call putsrt (xMod, 'TuberWt'     , TuberWt)
      call putsrt (xMod, 'LeafAr'      , LeafAr)
      call putsrt (xMod, 'LeafArPl'    , LeafArPl)
      call putsrt (xMod, 'TMSum'       , TmSum)
      call putsi  (xMod, 'LeafClCnt'   , LeafClCnt)
      call putsi  (xMod, 'LeafClOld'   , LeafClOld)
      call getarpm (xMod, 'UBGCalc0'    , UBGCalc0, 366, i, 0.)
   else if (xNewTask == 'do_rates') then
      if (LeafClCnt > 0) then
         do i = 1, LeafClCnt
            RLeafArLiveCl(i) = 0.
            RLeafArDeadCl(i) = 0.
            RLeafArLossCl(i) = 0.
            RLeafWtLiveCl(i) = 0.
            RLeafWtDeadCl(i) = 0.
            RLeafWtLossCl(i) = 0.
            RLeafAgeCl(i)    = 0.
         end do
      end if
      LeafWtClLiveTot = 0.
      LeafWtClDeadTot = 0.
      LeafWtClLossTot = 0.
      LeafWtGrowth    = 0.
      LeafArGrowth    = 0.
      LeafArGrowthPl  = 0.
      RLeafDeadWt     = 0.
      RLeafLossWt     = 0.
      RShufDMLeafLive = 0.
      RShufDMLeafDead = 0.
      StemWtGrowth    = 0.
      StemVolGrowth   = 0.
      RStemLiveWt     = 0.
      RStemDeadWt     = 0.
      RStemLossWt     = 0.
      RShufDMStemLive = 0.
      RShufDMStemDead = 0.
      TuberWtGrowth   = 0.
      RTuberLiveWt    = 0.
      RTuberDeadWt    = 0.
      RTuberLossWt    = 0.
      RShufDMTuberLive= 0.
      RShufDMTuberDead= 0.
      tuberrespiration= 0.
      TotWtGrowth     = 0.
      RootDepthChange = 0.
      RootHeightChange= 0.
      RConvLossReserve= 0.
      RConvLossRedistr= 0.
      FrReserveUse    = 0.
```

```
      FrRedistrUse    = 0.
      call getsi (xMod,'idoy',idoy)
      if (DateCmp2 < 1) then
         if (LeafClCnt > 0) then
            call getsrt (xMod, 'TMEff'          , TMEff)
            call getsrt (xMod, 'RDD'            , RDD)
            call getsrt (xMod, 'SourcePotP'       , SourcePotP)
            call getsrt (xMod, 'LeafArGrowthP'    , LeafArGrowthP)
            call getsrt (xMod, 'LeafArGrowthPlP'  , LeafArGrowthPlP)
            call getsrt (xMod, 'LeafWtGrowthP'    , LeafWtGrowthP)
            call getsrt (xMod, 'TuberWtGrowthP'   , TuberWtGrowthP)
            call getsrt (xMod, 'StemVolGrowthP'    , StemVolGrowthP)
            call getsrt (xMod, 'StemWtGrowthP'    , StemWtGrowthP)
            call getsrt (xMod, 'RootDepthChangeP' , RootDepthChangeP)
            call getsrt (xMod, 'RootHeightChangeP', RootHeightChangeP)
            call getsrtm(xMod, 'FrReserveUseP'    , FrReserveUseP, 0.)
            call getsrtm(xMod, 'FrRedistrUseP'    , FrRedistrUseP, 0.)
            call getsrtm(xMod, 'RedistrDMTot'     , RedistrDMTot, 0.)
            call getsrtm(xMod, 'ReserveDMTot'     , ReserveDMTot, 0.)
            call getsrtm(xMod, 'FreeDMLeafLive'   , FreeDMLeafLive, 0.)
            call getsrtm(xMod, 'FreeDMStemLive'   , FreeDMStemLive, 0.)
            call getsrtm(xMod, 'FreeDMTuberLive'  , FreeDMTuberLive,0.)
            call getsrt (xMod, 'StrucDMTuberLive'  , StrucDMTuberLive)
            call getsrt (xMod, 'FINT'    , FINT)
            call getsrt (xMod, 'FINTTOT' , FINTTOT)
            call getsrt (xMod, 'StressIndexLUE'   , StressIndexLUE )
            call getsrt (xMod, 'StressIndexLEff'  , StressIndexLEff)
            call getsrt (xMod, 'StressIndexLvAr'  , StressIndexLvAr)
            call getsrt (xMod, 'StressIndexStAg'  , StressIndexStAg)
            call getsrt (xMod, 'StressIndexRoAg'  , StressIndexRoAg)
            call getsrt (xMod, 'StressIndexTuAg'  , StressIndexTuAg)
            call getsrt (xMod, 'StressIndexTuGr'  , StressIndexTuGr)
            call getsrt (xMod, 'StressIndexStAgP' , StressIndexStAgP)
            call getsrt (xMod, 'StressIndexRoAgP' , StressIndexRoAgP)
            call getsrt (xMod, 'StressIndexTuAgP' , StressIndexTuAgP)
            call getsrt (xMod, 'StressIndexLvWt'  , StressIndexLvWt)
            call getsrt (xMod, 'StressIndexStWt'  , StressIndexStWt)
            call getsrt (xMod, 'StressIndexRtWt'  , StressIndexRtWt)
            call getsrt (xMod, 'StressIndexTuWt'  , StressIndexTuWt)
            call getsrt (xMod, 'StressIndexRtDp'  , StressIndexRtDp)
            call getsrt (xMod, 'StressIndexRtHt'  , StressIndexRtHt)
            call getsrt (xMod, 'StressIndexTmSu'  , StressIndexTmSu)
            LeafArGrowthPl   = LeafArGrowthPlP * StressIndexLvAr
            LeafArGrowth     = max(0.,LeafArGrowthPl * NPL)
            call GetSLA(SLA)
            LeafWtGrowth     = (LeafArGrowth / 10000.) / max(1.e-6, SLA)
            StemVolGrowth    = StemVolGrowthP * StressIndexStWt
            StemWtGrowth     = StemWtGrowthP  * StressIndexStWt
            AboveDryWtGrowth = LeafWtGrowth + StemWtGrowth
            AboveGlucoseUse  = LeafWtGrowth * GVILeaves + StemWtGrowth * &
                               GVIStems
            SourcePot        = SourcePotP * StressIndexLue
            FrReserveUse     = 0.
            RConvLossReserve = 0.
            FrRedistrUse     = 0.
            RConvLossRedistr = 0.
            RedistrLeafDead = 0.
            ReserveLeafLive = 0.
            do i = 1, leafclcnt
               if ( (not(LeafAliveCl(i))) .and. (FreeDMLeafCl(i) >0.))then
                  RedistrLeafDead    = RedistrLeafDead + FreeDMleafCl(i)
               elseif (LeafAliveCl(i)) then
                  ReserveLeafLive  = ReserveLeafLive + FreeDMleafCl(i) * &
                    kdestableaf
               end if
            end do
            ReserveStemLive  = max(0., 1. - (StressIndexStAg + &
               StressIndexStAgP)) * FreeDMStemLive  * kdestabstem
```

```
338            ReserveTuberLive = max(0., 1. - (StressIndexTuAg + &
339               StressIndexTuAgP)) * FreeDMTuberLive * kdestabtuber
340            RedistrStemDead  = min(1.,StressIndexStAg+StressIndexStAgP) &
341               * FreeDMStemLive
342            RedistrTuberDead = min(1.,StressIndexTuAg+ StressIndexTuAgP) &
343               * FreeDMTuberLive
344            RedistrDMTot =RedistrLeafDead+RedistrStemDead+RedistrTuberDead
345            ReserveDMTot =ReserveLeafLive+ReserveStemLive+ReserveTuberLive
346            if (SourcePot < AboveGlucoseUse) then
347               FrRedistrUse     = min(1., max(0., (AboveGlucoseUse - &
348                  SourcePot ) / max(1.e-10, RedistrDMTot * GVIReserves * &
349                  ConvReserveDM)))
350               if (SourcePot + FrRedistrUse * RedistrDMTot * GVIReserves &
351                  * ConvReserveDM < AboveGlucoseUse) then
352                  FrReserveUse     = min(1., max(0., (AboveGlucoseUse - &
353                     SourcePot - FrRedistrUse * RedistrDMTot*GVIReserves&
354                     * ConvReserveDM) / & max(1.e-10, ReserveDMTot * &
355                     GVIReserves * ConvReserveDM)))
356               end if
357               RelGrowth= max(0.,(SourcePot+ GVIReserves * ConvReserveDM &
358                 * ( (FrRedistrUse * RedistrDMTot) + (FrReserveUse * &
359                 ReserveDMTot) ) ) / max(1.e-6, AboveGlucoseUse))
360               LeafArGrowthPl    = LeafArGrowthPl * RelGrowth
361               LeafArGrowth      = LeafArGrowthPl * NPL
362               LeafWtGrowth      = LeafWtGrowth * RelGrowth
363               StemVolGrowth     = RelGrowth * StemVolGrowth
364               StemWtGrowth      = RelGrowth * StemWtGrowth
365               AboveDryWtGrowth  = LeafWtGrowth + StemWtGrowth
366               AboveGlucoseUse   = LeafWtGrowth * GVILeaves+StemWtGrowth &
367                  * GVIStems
368            end if
369            RConvLossRedistr = FrRedistrUse * RedistrDMTot * ((1. - &
370               ConvReserveDM) + (1.- GVIReserves) * ConvReserveDM)
371            RConvLossReserve = FrReserveUse * ReserveDMTot * ((1. - &
372               ConvReserveDM) + (1.- GVIReserves) * ConvReserveDM)
373            TuberWtGrowth = StressIndexTuGr * max(0., (SourcePot - &
374               AboveGlucoseUse + ( GVIReserves * FrRedistrUse * &
375               RedistrDMTot * ConvReserveDM ) + ( GVIReserves * &
376               FrReserveUse * ReserveDMTot * ConvReserveDM ) )/GVITubers )
377            RootDepthChange  = RootDepthChangeP  * StressIndexRtDp
378            RootHeightChange = RootHeightChangeP * StressIndexRtHt
379            TotWtGrowth      = LeafWtGrowth + StemWtGrowth + TuberWtGrowth
380            LeafAr = 0.
381            LeafClOldNew = LeafClOld
382            RLeafLiveAr  = 0.
383            RLeafDeadAr  = 0.
384            RLeafLossAr  = 0.
385            RLeafLiveWt  = LeafWtGrowth
386            RLeafDeadWt  = 0.
387            RLeafLossWt  = 0.
388            RStemLiveWt  = 0.
389            RStemDeadWt  = 0.
390            RStemLossWt  = 0.
391            RTuberLiveWt = 0.
392            RTuberDeadWt = 0.
393            RTuberLossWt = 0.
394            do i = LeafClCnt, 1,-1
395               LeafAge = LeafAgeCl(i)+RLeafAgeClP(i)*StressIndexLvAgCl(i)
396               RLeafAgeCl(i) = LeafAge - LeafAgeCL(i)
397               RLeafWtLiveCl(i) = 0.
398               RLeafWtDeadCl(i) = 0.
399               RLeafWtLossCl(i) = 0.
400               RLeafArLiveCl(i) = 0.
401               RLeafArDeadCl(i) = 0.
402               RLeafArLossCl(i) = 0.
403               FrSenescence(i)  = 0.
404               if ((not(LeafAliveCl(i))).and.(FreeDMLeafCl(i) > 0.)) then
405                  RLeafWtDeadCl(i) =   LeafWtLiveCl(i) - FrRedistrUse * &
```

```
                  FreeDMLeafCl(i)
               RLeafWtLiveCl(i) = - LeafWtLiveCl(i)
               RLeafArDeadCl(i) =   LeafArLiveCl(i)
               RLeafArLiveCl(i) = - LeafArLiveCl(i)
               FrSenescence(i)  = 1.
            elseif (not(LeafAliveCl(i))) then
               RLeafWtDeadCl(i) = - LeafWtDeadCl(i) * RelLossDeadLeaves
               RLeafWtLossCl(i) =   LeafWtDeadCl(i) * RelLossDeadLeaves
               RLeafArDeadCl(i) = - LeafArDeadCl(i) * RelLossDeadLeaves
               RLeafArLossCl(i) =   LeafArDeadCl(i) * RelLossDeadLeaves
            else
               RLeafWtDeadCl(i) =   (LeafWtLiveCl(i) - FrReserveUse * &
                  FreeDMLeafCl(i)*kdestableaf)*StressIndexLvDyingCl(i)
               RLeafWtLiveCl(i) = - FrReserveUse * FreeDMLeafCl(i) * &
                  kdestableaf - RLeafWtDeadCl(i)
               RLeafArDeadCl(i)=LeafArLiveCl(i)*StressIndexLvDyingCl(i)
               RLeafArLiveCl(i) = - RLeafArDeadCl(i)
            end if
            RLeafLiveWt = RLeafLiveWt + RLeafWtLiveCl(i)
            RLeafDeadWt = RLeafDeadWt + RLeafWtDeadCl(i)
            RLeafLossWt = RLeafLossWt + RLeafWtLossCl(i)
            RLeafLiveAr = RLeafLiveAr + RLeafArLiveCl(i)
            RLeafDeadAr = RLeafDeadAr + RLeafArDeadCl(i)
            RLeafLossAr = RLeafLossAr + RLeafArLossCl(i)
         end do
         call putsi(xMod, 'LeafClOldNew', LeafClOldNew)
         RStemLiveWt  = StemWtGrowth - FrReserveUse * ReserveStemLive &
            - min(1., StressIndexStAg + StressIndexStAgP) * (StemWt - &
            FrReserveUse * ReserveStemLive)
         RTuberLiveWt = TuberWtGrowth - FrReserveUse*ReserveTuberLive&
            - min(1., StressIndexTuAg + StressIndexTuAgP) * (TuberWt- &
            FrReserveUse * ReserveTuberLive)
         TuberRespiration = max(0., StrucDMTuberLive  * RelRespTuber )
         RStemDeadWt  = min(1., StressIndexStAg + StressIndexStAgP) * &
            (StemWt  - FrReserveUse * ReserveStemLive) - FrRedistrUse &
            * RedistrStemDead  - StemDeadWt  * RelLossDeadStems
         RTuberDeadWt = min(1., StressIndexTuAg + StressIndexTuAgP) * &
            (TuberWt - FrReserveUse * ReserveTuberLive) - FrRedistrUse&
            * RedistrTuberDead - TuberDeadWt * RelLossDeadTubers
         RStemLossWt  = StemDeadWt  * RelLossDeadStems
         RTuberLossWt = TuberDeadWt * RelLossDeadTubers
      end if
   elseif (DateCmp3 < 1) then
      TuberRespiration = max(0., StrucDMTuberLive  * RelRespTuber )
   end if
   call getSLA(SLA)
   call putsrt (xMod, 'TuberRespiration', TuberRespiration)
   call putsrt (xMod, 'FrReserveUse'    , FrReserveUse)
   call putsrt (xMod, 'FrRedistrUse'    , FrRedistrUse)
   call putsrt (xMod, 'ReserveLeafLive' , ReserveLeafLive)
   call putsrt (xMod, 'ReserveStemLive' , ReserveStemLive)
   call putsrt (xMod, 'ReserveTuberLive', ReserveTuberLive)
   call putsrt (xMod, 'RedistrLeafDead' , RedistrLeafDead)
   call putsrt (xMod, 'RedistrStemDead' , RedistrStemDead)
   call putsrt (xMod, 'RedistrTuberDead', RedistrTuberDead)
   call putsrt (xMod, 'LeafWtGrowth'    , LeafWtGrowth)
   call putsrt (xMod, 'LeafArGrowth'    , LeafArGrowth)
   call putsrt (xMod, 'TuberWtGrowth'   , TuberWtGrowth)
   call putsrt (xMod, 'StemWtGrowth'    , StemWtGrowth)
   call putsrt (xMod, 'StemVolGrowth'   , StemVolGrowth)
   call putsrt (xMod, 'RLeafDeadWt'     , RLeafDeadWt)
   call putsrt (xMod, 'RStemDeadWt'     , RStemDeadWt)
   call putsrt (xMod, 'RTuberDeadWt'    , RTuberDeadWt)
     else if (xNewTask == 'output') then
   call getsrtm(xMod, 'RShufDMLeafLive',    RShufDMLeafLive, 0.)
   call getsrtm(xMod, 'RShufDMStemLive',    RShufDMStemLive, 0.)
   call getsrtm(xMod, 'RShufDMTuberLive',   RShufDMTuberLive, 0.)
   call getsrtm(xMod, 'RShufDMLeafDead',    RShufDMLeafDead, 0.)
```

```
474      call getsrtm(xMod, 'RShufDMStemDead',    RShufDMStemDead, 0.)
475      call getsrtm(xMod, 'RShufDMTuberDead',   RShufDMTuberDead, 0.)
476      RshufDMTot = ConvReserveDM * GVIReserves * (RShufDMLeafLive + &
477         RShufDMStemLive + RShufDMTuberLive +RShufDMLeafDead + &
478         RShufDMStemDead + RShufDMTuberDead )
479      RConvLossRedistr = RconvLossRedistr + ((1. - ConvReserveDM) + &
480         (1.- GVIReserves) * ConvReserveDM) * ( RShufDMLeafDead + &
481         RShufDMStemDead + RShufDMTuberDead )
482      RConvLossReserve = RconvLossReserve + ((1. - ConvReserveDM) + &
483        (1.- GVIReserves) * ConvReserveDM) * ( RShufDMLeafLive +  &
484        RShufDMStemLive + RShufDMTuberLive)
485      call outdat (2,0,'LAILive'        , LAILive)
486      call outdat (2,0,'LAIDead'        , LAIDead)
487      call outdat (2,0,'LAILoss'        , LAILoss)
488      call outdat (2,0,'StemWt'         , StemWt)
489      call outdat (2,0,'StemPresentWt'  , StemPresentWt)
490      call outdat (2,0,'StemDeadWt'     , StemDeadWt)
491      call outdat (2,0,'TuberWt'        , TuberWt)
492      call outdat (2,0,'TuberFreshWt'   , TuberFreshWt)
493      call outdat (2,0,'OWGCalc'        , OWGCalc)
494      call outdat (2,0,'UBGCalc'        , UBGCalc)
495      call outdat (2,0,'StarchCalc'        , StarchCalc)
496   else if (xNewTask == 'do_states') then
497      call getsi (xMod,'idoy',idoy)
498      call getsi (xMod,'iyear',iyear)
499      call getsim (xMod, 'CropStYear',      CropStYear,      0)
500      call getsim (xMod, 'CropStDay',       CropStDay,       0)
501      if (CropStYear <= 0) then
502         CropStYear =iyear
503         call cmdelvar('CropStYear')
504         call putsi (xMod, 'CropStYear', CropStYear)
505      end if
506      if (CropStDay <= 0) then
507         CropStDay = nint(Fintim)
508         call cmdelvar ('CropStDay')
509         call putsi (xMod, 'CropStDay', CropStDay)
510      end if
511      DateCmp  = dtfsecmp (CropStYear,      CropStDay,      iyear, idoy)
512      DateCmp2 = dtfsecmp (LaiLeafKillYear, LaiLeafKillDay, iyear, idoy)
513      DateCmp3 = dtfsecmp (CropHarvestYear, CropHarvestDay, iyear, idoy)
514      call putsi (xMod, 'DateCmp', DateCmp)
515      call putsi (xMod, 'DateCmp2', DateCmp2)
516      call putsi (xMod, 'DateCmp3', DateCmp3)
517      if (DateCmp2 < 1) then
518         if (Delt < 1.0) call fatalerr (xMod,'Delt too small')
519         if (DateCmp ==  0) then
520            call getsrp  (xMod,'NPL'       , NPL)
521            LeafArPl        = LeafArPlI
522            LAILive         = LeafArPl*NPL/10000.
523            LAIEff          = LAILive
524            LaiDead         = 0.
525            LaiLoss         = 0.
526            LAITot          = LAILive + LaiDead
527            LAIMx           = LAITot
528            TMSum           = TMSumI
529            LeafAr          = LAILive*10000.
530            LeafWt          = LAILive/max(1.e-6, SLA)
531            StemWt          = StemWtI
532            TuberWt         = TuberWtI
533            LeafDeadWt      = LeafDeadWtI
534            StemDeadWt      = StemDeadWtI
535            TuberDeadWt     = TuberDeadWtI
536            TotalDryWt      = TuberWt + StemWt + LeafWt + LeafDeadWt + &
537               StemDeadWt + TuberDeadWt
538            TotalDryWtI     = TotalDryWt
539            AboveDryWt      = StemWt +LeafWt
540            TotalAboveDryWt = StemWt +LeafWt+LeafDeadWt + StemDeadWt
541            HarvestIndex = TuberWt / max(1.e-6, TuberWt + StemWt + LeafWt)
```

```
            LeafClCnt       = LeafClCntI
            if (LeafClCnt == 1) then
               LeafArLiveCl(leafClCnt) = 0.
               LeafArDeadCl(leafClCnt) = 0.
               LeafArLossCl(leafClCnt) = 0.
               RLeafArLiveCl(LeafClCnt) = LeafAr * delt
               RLeafArDeadCl(LeafClCnt) = 0.
               RLeafArLossCl(LeafClCnt) = 0.
               LeafWtLiveCl(leafClCnt) = 0.
               LeafWtDeadCl(leafClCnt) = 0.
               LeafWtLossCl(leafClCnt) = 0.
               RLeafWtLiveCl(LeafClCnt) = LeafWt * delt
               RLeafWtDeadCl(LeafClCnt) = 0.
               RLeafWtLossCl(LeafClCnt) = 0.
               LeafAgeCl(LeafClCnt)   = 0.
               LeafAliveCl(LeafClCnt) = .true.
            else
               call fatalerr (xMod, 'LeafClCntI /= 1 not yet implemented')
            end if
            RootDepth       = RootDepthI
            RootHeight      = RootHeightI
         else if(DateCmp > 0) then
            TMSum = intgrl (TMSum, TMEff * StressIndexTmSu , Delt)
            if (LeafArGrowth > 0.) then
               LeafClCnt = LeafClCnt + 1
               if (LeafClCnt > LeafClMxn) call fatalerr &
                     (xMod,'too many leaf classes')
               LeafArLiveCl(leafClCnt) = 0.
               LeafArDeadCl(leafClCnt) = 0.
               LeafArLossCl(leafClCnt) = 0.
               RLeafArLiveCl(LeafClCnt) = LeafArGrowth * delt
               RLeafArDeadCl(LeafClCnt) = 0.
               RLeafArLossCl(LeafClCnt) = 0.
               LeafWtLiveCl(leafClCnt) = 0.
               LeafWtDeadCl(leafClCnt) = 0.
               LeafWtLossCl(leafClCnt) = 0.
               RLeafWtLiveCl(LeafClCnt) = LeafWtGrowth * delt
               RLeafWtDeadCl(LeafClCnt) = 0.
               RLeafWtLossCl(LeafClCnt) = 0.
               LeafAgeCl(LeafClCnt)   = 0.
               LeafAliveCl(LeafClCnt) = .true.
            end if
         end if
         LeafAr = 0.
         LeafArDead = 0.
         LeafArLoss = 0.
         LeafClOld = 0
         LeafWtClLiveTot = 0.
         LeafWtClDeadTot = 0.
         LeafWtClLossTot = 0.
         if (LeafClCnt > 0) then
            do i = 1, LeafClCnt
               if(LeafArLiveCl(i) + RLeafArLiveCl(i) > 1.e-8 ) then
                  LeafArLiveCl(i) = intgrl(LeafArLiveCl(i), &
                     RLeafArLiveCl(i), delt)
               else
                  LeafArLiveCl(i) = 0.
               end if
               if(LeafArDeadCl(i) + RLeafArDeadCl(i) > 1.e-8 ) then
                  LeafArDeadCl(i) = intgrl(LeafArDeadCl(i), &
                     RLeafArDeadCl(i), delt)
               else
                  LeafArDeadCl(i) = 0.
               end if
               if(LeafArLossCl(i) + RLeafArLossCl(i) > 1.e-8 ) then
                  LeafArLossCl(i) = intgrl(LeafArLossCl(i), &
                     RLeafArLossCl(i), delt)
               else
```

```
            LeafArLossCl(i) = 0.
         end if
         if(LeafWtLiveCl(i) + RLeafWtLiveCl(i) > 1.e-8 ) then
            LeafWtLiveCl(i) = intgrl(LeafWtLiveCl(i), &
               RLeafWtLiveCl(i), delt)
         else
            LeafWtLiveCl(i) = 0.
         end if
         if(LeafWtDeadCl(i) + RLeafWtDeadCl(i) > 1.e-8 ) then
            LeafWtDeadCl(i) = intgrl(LeafWtDeadCl(i), &
               RLeafWtDeadCl(i), delt)
         else
            LeafWtDeadCl(i) = 0.
         end if
         if(LeafWtLossCl(i) + RLeafWtLossCl(i) > 1.e-8 ) then
            LeafWtLossCl(i) = intgrl(LeafWtLossCl(i), &
               RLeafWtLossCl(i), delt)
         else
            LeafWtLossCl(i) = 0.
         end if
         LeafWtClLiveTot = LeafWtClLiveTot + LeafWtLiveCl(i)
         LeafWtClDeadTot = LeafWtClDeadTot + LeafWtDeadCl(i)
         LeafWtClLossTot = LeafWtClLossTot + LeafWtLossCl(i)
         LeafAr          = LeafAr          + LeafArLiveCl(i)
         LeafArDead      = LeafArDead      + LeafArDeadCl(i)
         LeafArLoss      = LeafArLoss      + LeafArLossCl(i)
         LeafAgeCl(i)    = intgrl(LeafAgeCl(i), RLeafAgeCl(i), delt)
      end do
      LeafArPl  = intgrl (LeafArPl , LeafArGrowthPl , delt)
      StemVol   = intgrl (StemVol  , StemVolGrowth - min(1., &
         StressIndexStAg + StressIndexStAgP) * StemVol, delt)
      LeafWt     = LeafWtClLiveTot
      LeafDeadWt = LeafWtClDeadTot
      LeafLossWt = LeafWtClLossTot
      StemWt = intgrl (StemWt, RStemLiveWt - RShufDMStemLive , delt)
      StemDeadWt = intgrl(StemDeadWt, StemDeadWt - RShufDMStemDead &
         , delt)
      StemLossWt    = intgrl (StemLossWt  , RStemLossWt, delt)
      TuberWt = intgrl(TuberWt, RTuberLiveWt - RShufDMTuberLive - &
         TuberRespiration, delt)
      TuberDeadWt = intgrl (TuberDeadWt , RTuberDeadWt - &
         RShufDMTuberDead, delt)
      TuberLossWt   = intgrl (TuberLossWt , RTuberLossWt, delt)
      LeafPresentWt = LeafWt + LeafDeadWt
      StemPresentWt = StemWt + StemDeadWt
      AbovePresentWt = LeafPresentWt + StemPresentWt
      RootDepth = intgrl (RootDepth, RootDepthChange, delt)
      RootHeight = intgrl (RootHeight, RootHeightChange, delt)
      TParInt   = intgrl (TParInt, (ParFraction*RDD*FINT), Delt)
      SourcePotCum = intgrl (SourcePotCum, TotWtGrowth , Delt)
      TuberRespirationCum = intgrl (TuberRespirationCum, &
         TuberRespiration, Delt)
      RedistrDMTotCum = intgrl (RedistrDMTotCum, FrRedistrUse * &
         RedistrDMTot, delt)
      ReserveDMTotCum = intgrl (ReserveDMTotCum, FrReserveUse * &
         ReserveDMTot, delt)
      ConvLossReserve = intgrl (ConvLossReserve, RConvLossReserve, &
         delt)
      ConvLossRedistr = intgrl (ConvLossRedistr, RConvLossRedistr, &
         delt)
      SLARTot       = LAITot / max(1.e-10, LeafWt + LeafDeadWt)
      HarvestIndex = TuberWt / max(1.e-6,TuberWt + StemWt + LeafWt)
      TotalDryWt   = TuberWt + StemWt + LeafWt + LeafDeadWt + &
         StemDeadWt + TuberDeadWt
      AboveDryWt      = StemWt + LeafWt
      TotalAboveDryWt = StemWt + LeafWt + LeafDeadWt + StemDeadWt
      LiveDryWt       = AboveDryWt + TuberWt
      NonTuberWt      = AboveDryWt
```

```
678            if (tuberwt > 1.e-8) then
679               OWGCalc       = alphaOWG        * (tuberwt**betaOWG)
680               TuberFreshWt  = alphaTuberFresh * (tuberwt**betaTuberFresh)
681               UBGCalc       = alphaUBG        * (tuberwt**betaUBG)
682               StarchCalc    = alphaStarch     * (tuberwt**betaStarch)
683            else
684               TuberFreshWt  = 0.
685               OWGcalc       = 0.
686               UBGcalc       = 0.
687               Starchcalc    = 0.
688            end if
689         end if
690         if (leafclcnt > 0) then
691             call putsi (xMod, 'leafclold', leafclold)
692         endif
693         if (DateCmp2 ==  0) then
694            laileafkill = finttot
695            LeafWtClLiveTot = 0.
696            LeafWtClDeadTot = 0.
697            LeafWtClLossTot = 0.
698            LeafAr          = 0.
699            LeafArDead      = 0.
700            LeafArLoss      = 0.
701            if (leafclcnt > 0) then
702               do i = 1, LeafClCnt
703                  RLeafArLiveCl(i) = - LeafArLiveCl(i)
704                  LeafArLiveCl(i) = intgrl(LeafArLiveCl(i), &
705                     RLeafArLiveCl(i), delt)
706                  RLeafArDeadCl(i) = -LeafArDeadCl(i)
707                  LeafArDeadCl(i) = intgrl(LeafArDeadCl(i), &
708                     RLeafArDeadCl(i), delt)
709                  RLeafArLossCl(i) = -RleafArLiveCl(i) - RLeafArDeadCl(i)
710                  LeafArLossCl(i) = intgrl(LeafArLossCl(i), &
711                     RLeafArLossCl(i), delt)
712                  RLeafWtLiveCl(i) = - LeafWtLiveCl(i)
713                  LeafWtLiveCl(i) = intgrl(LeafWtLiveCl(i), &
714                     RLeafWtLiveCl(i), delt)
715                  RLeafWtDeadCl(i) = - LeafWtDeadCl(i)
716                  LeafWtDeadCl(i) = intgrl(LeafWtDeadCl(i), &
717                     RLeafWtDeadCl(i), delt)
718                  RLeafWtLossCl(i) = -RleafWtLiveCl(i) - RLeafWtDeadCl(i)
719                  LeafWtLossCl(i) = intgrl(LeafWtLossCl(i), &
720                     RLeafWtLossCl(i), delt)
721                  LeafWtClLiveTot = LeafWtClLiveTot + LeafWtLiveCl(i)
722                  LeafWtClDeadTot = LeafWtClDeadTot + LeafWtDeadCl(i)
723                  LeafWtClLossTot = LeafWtClLossTot + LeafWtLossCl(i)
724                  LeafAr          = LeafAr          + LeafArLiveCl(i)
725                  LeafArDead      = LeafArDead      + LeafArDeadCl(i)
726                  LeafArLoss      = LeafArLoss      + LeafArLossCl(i)
727               end do
728               leafclcnt = 0
729            end if
730            RStemLiveWt = -StemWt
731            RStemDeadWt = -StemDeadWt
732            RStemLossWt = -RStemLiveWt - RStemDeadWt
733            StemWt        = intgrl (StemWt      , RStemLiveWt, delt)
734            StemDeadWt    = intgrl (StemDeadWt  , RStemDeadWt, delt)
735            StemLossWt    = intgrl (StemLossWt  , RStemLossWt, delt)
736            StemPresentWt = StemWt + StemDeadWt
737            LeafPresentWt = LeafWt + LeafDeadWt
738            StemPresentWt = StemWt + StemDeadWt
739            AbovePresentWt = LeafPresentWt + StemPresentWt
740            TotalDryWt      = TuberWt + StemWt + LeafWt + LeafDeadWt + &
741               StemDeadWt + TuberDeadWt
742            AboveDryWt      = StemWt + LeafWt
743            TotalAboveDryWt = StemWt + LeafWt + LeafDeadWt + StemDeadWt
744            LiveDryWt       = AboveDryWt + TuberWt
745            NonTuberWt      = AboveDryWt
```

```
      end if
      LAILive = LeafAr / 10000.
      LaiEff  = LaiLive
      LAIDead = LeafArDead / 10000.
      LAILoss = LeafArLoss / 10000.
      LAITot  = LAILive + LAIDead
      if (LAILive > LAIMx) then
         DayLaiMx = 1.*idoy
         LAIMx = LAITot
      end if
   elseif (DateCmp3 < 1) then
      TuberWt             = intgrl (TuberWt, -TuberRespiration, delt)
      TuberRespirationCum = intgrl (TuberRespirationCum, &
         TuberRespiration, Delt)
      TotalDryWt      = TuberWt + StemWt + LeafWt + LeafDeadWt + &
         StemDeadWt + TuberDeadWt
      LiveDryWt       = AboveDryWt + TuberWt
   end if
   DMIncoming  = SourcePotCum + TotalDryWtI - (ReserveDmTotCum + &
      RedistrDMTotCum)
   DMAccounted = TotalDryWt + LeafLossWt + StemLossWt + TuberLossWt + &
      TuberRespirationCum
   DMBalance = DmIncoming - DMAccounted
   if(abs(DMBalance)/max(1.e-10,DMIncoming + DMAccounted) > 1.e-6) then
      write(*,*) DMbalance, DMIncoming, DMAccounted
   end if
   call putsrt (xMod, 'LaiLive'     , laiLive)
   call putsrt (xMod, 'LaiMx'       , laiMx)
   call putsrt (xMod, 'DayLaiMx'    , DaylaiMx)
   call putsrt (xMod, 'LaiLeafKill' , LaiLeafKill)
   call putsrt (xMod, 'LaiDead'     , laiDead)
   call putsrt (xMod, 'LaiLoss'     , laiLoss)
   call putsrt (xMod, 'LaiTot'      , laiTot)
   call putsrt (xMod, 'laiEff'      , laiEff)
   call putsi  (xMod, 'leafclcnt', leafclcnt)
   call putsrt (xMod, 'TuberFreshWt', TuberFreshWt)
   call putsrt (xMod, 'OWGCalc'     , OWGCalc)
   call putsrt (xMod, 'UBGCalc'     , UBGCalc)
   call putsrt (xMod, 'StarchCalc'  , StarchCalc)
   call putsrt (xMod, 'RootDepth'   , RootDepth)
   call putsrt (xMod, 'RootHeight'  , RootHeight)
   call putsrt (xMod, 'LeafWt'      , LeafWt)
   call putsrt (xMod, 'LeafPresentWt', LeafPresentWt)
   call putsrt (xMod, 'LeafDeadWt'  , LeafDeadWt)
   call putsrt (xMod, 'LeafLossWt'  , LeafLossWt)
   call putsrt (xMod, 'StemWt'      , StemWt)
   call putsrt (xMod, 'StemDeadWt'  , StemDeadWt)
   call putsrt (xMod, 'StemLossWt'  , StemLossWt)
   call putsrt (xMod, 'StemPresentWt', StemPresentWt)
   call putsrt (xMod, 'TuberWt'     , TuberWt)
   call putsrt (xMod, 'TuberDeadWt'  , TuberDeadWt)
   call putsrt (xMod, 'AboveDryWt'   , AboveDryWt)
   call putsrt (xMod, 'AbovePresentWt'  , AbovePresentWt)
   call putsrt (xMod, 'LiveDryWt'   , LiveDryWt)
   call putsrt (xMod, 'LeafAr'      , LeafAr)
   call putsrt (xMod, 'LeafArPl'    , LeafArPl)
   call putsrt (xMod, 'TMSum'       , TMSum)
   call putsrt (xMod, 'NonTuberWt'  , NonTuberWt)
   call putsrt (xMod, 'StemVol'     , StemVol)
   call putsrt (xMod, 'TotalDryWt'  , TotalDryWt)
   call putsrt (xMod, 'SLARTot'     , SLARTOT)
   call writesrtatdebug(xlogfileunit, xMod, "states", 1, namestates)
end if
end subroutine Lupt3bRea
```

I.21. Subroutine MANAGEMENT

```
subroutine Management (xNewTask, xMod, xLogFileUnit)
implicit none
integer, parameter :: TempUnit=50,MaxNrManage=1000,MaxNrFertChar = 10,&
   MaxNrPestChar = 10, MaxNrChar = max (MaxNrFertChar, MaxNrPestChar),&
   MaxNrPesticides = 50
integer :: i, ii, iii, NrManage, IYear, IDoy, NrFertChar, NrPestChar, &
   CIndex, NIndex, PIndex, KIndex, DMIndex, MinNIndex, NrPesticides, &
   IPesticide, IP, managestartyear,managestartday, xLogFileUnit
integer, dimension(MaxNrManage) :: ManageYear, ManageDay
real, parameter :: tiny2 = 0.01
real, dimension (MaxNrManage) :: ManageInAmount, ManageDepth
real, dimension (MaxNrFertChar) :: FertChar
real, dimension (MaxNrPestChar) :: PestChar
real, dimension (MaxNrPesticides) :: PestChars
real, dimension (MaxNrFertChar) :: TotNan, TotNor
real, dimension (MaxNrPesticides) :: PesticideAmount, &
   PesticideAmountCum
real, dimension (MaxNrPestChar, MaxNrPesticides) :: PestCharUsed
real :: irrigation, irrigationicasa, pldep, fertdepth, watermanure, &
   FrMinNOrg, EmFractEst
CHARACTER*(*) :: xNewTask, xMod
character(len=13) :: OldTask, EmptyString
character(len=80) :: ManagementDataFile, FertilizerDataFile, &
   PesticideDataFile
character(len=250) :: HeaderString
character(len=100), dimension (MaxNrManage) :: ManageType, &
   ManageInType, ManageInDim
character(len=100) :: FertName
character(len=100), dimension (MaxNrFertChar) :: FertCharNames
character(len=100), dimension (MaxNrPestChar) :: PestCharNames
character(len=100) :: PestName
character(len=100), dimension (MaxNrPesticides) ::PesticidesNamesUsed,&
   PesticideNames
character*80, parameter :: Emptyfile = 'nn'
character*30 :: soilactivity
logical :: DoBlightStress, Dosoilblight, DoWaterStress, DoNitroStress,&
   icmode
logical, dimension(MaxNrManage) :: ManageDone
save
if (xNewTask == 'initialize') then
   call getslm (xMod, 'DoBlightStress', DoBlightStress, .false.)
   call getslm(xMod, 'Dosoilblight'      , Dosoilblight, .false.)
   call getslm (xMod, 'DoWaterStress', DoWaterStress, .false.)
   call getslm (xMod, 'DoNitroStress', DoNitroStress, .false.)
   call getsc (xMod, 'ManagementDataFile', ManagementDataFile)
   call getsc (xMod, 'FertilizerDataFile', FertilizerDataFile)
   call getsc (xMod, 'PesticideDataFile' , PesticideDataFile )
   call getsl (xMod, 'icmode', icmode)
   if (managementdatafile.eq.Emptyfile) then
      if (ICMODE) then
         call fatalerr (xMod, &
         'Please define the managementdatafile in the PTS-file')
      else
         call fatalerr (xMod, &
         'Please define the managementdatafile in the COMBCTL-file')
      end if
   else
      open (Unit = TempUnit, file = ManagementDataFile, status = 'old')
      i = 1
      read (TempUnit, '(A)') HeaderString
1           read (TempUnit, *, err = 2, end = 2) ManageYear(i), &
               ManageDay(i), ManageType(i), ManageInType(i), &
               ManageInAmount(i), ManageDepth(i), ManageInDim(i)
      call Lowerc(ManageType(i))
      call Lowerc(ManageInType(i))
```

```
66          ManageDone(i) = .false.
67          i = i + 1
68          goto 1
69    2     NrManage = i - 1
70          close (TempUnit)
71       endif
72       if ((DoNitroStress) .or. (DoWaterStress)) then
73          if (fertilizerdatafile /= Emptyfile) then
74             open (Unit=TempUnit,File = FertilizerDataFile, status = 'old')
75             read (TempUnit, '(A)') HeaderString
76             call FindNrEntries(HeaderString, NrFertChar, FertCharNames, &
77                MaxNrFertChar)
78             close (TempUnit)
79             CIndex = 0
80             NIndex = 0
81             PIndex = 0
82             KIndex = 0
83             DMIndex = 0
84             MinNIndex = 0
85             Do i = 2, NrFertChar+1
86                if (trim(adjustl(FertCharNames(i))) == 'C') then
87                   CIndex = i - 1
88                else if (trim(adjustl(FertCharNames(i))) == 'N') then
89                   NIndex = i - 1
90                else if (trim(adjustl(FertCharNames(i))) == 'P') then
91                   PIndex = i - 1
92                else if (trim(adjustl(FertCharNames(i))) == 'K') then
93                   KIndex = i - 1
94                else if (trim(adjustl(FertCharNames(i))) == 'DM') then
95                   DMIndex = i - 1
96                else if (trim(adjustl(FertCharNames(i))) == 'MinN') then
97                   MinNIndex = i - 1
98                else
99                   call fatalerr (xMod, 'Unknown characteristic in &
100                     Fertilizer.Dat')
101               end if
102            end do
103            if (CIndex*NIndex * PIndex * KIndex * DMIndex * MinNIndex == &
104               0) then
105               call fatalerr (xMod, 'Not all nutrients (C,N,P,K,DM, &
106                  MinN) are set in Fertilizer.Dat')
107            end if
108         end if
109      end if
110      if (pesticidedatafile /= Emptyfile) then
111         open (Unit = TempUnit, File = PesticideDataFile, status = 'old')
112         read (TempUnit, '(A)') HeaderString
113         call FindNrEntries(HeaderString, NrPestChar, PestCharNames, &
114            MaxNrPestChar)
115         close (TempUnit)
116      endif
117      NrPesticides = 0
118      do ii = 1, MaxNrPesticides
119         PesticidesNamesUsed(ii) = ''
120      end do
121      Ipesticide = 0
122      fertdepth = 0.
123      pldep = -99.
124      do ii = 1, NrFertChar
125         totnan(ii) = 0.
126         totnor(ii) = 0.
127      end do
128      watermanure = 0.
129      irrigation = 0.
130   else if (xNewTask == 'do_rates') then
131      call getsi (xMod, 'iyear', IYear)
132      call getsi (xMod, 'idoy',  IDoy)
133      if (doNitroStress) then
```

```
      call getsrp (xMod, 'FrMinNOrg', FrMinNOrg)
   end if
   Ipesticide = 0
   fertdepth = 0.
   pldep = -99.
    if ((doNitroStress) .or. (doWaterStress)) then
      do ii = 1, NrFertChar
         totnan(ii) = 0.
         totnor(ii) = 0.
      end do
      watermanure = 0.
   end if
   irrigation = 0.
   soilactivity = 'nn'
   do i = 1, NrManage
      if (ManageDone(i) == .false. ) then
         if((Iyear==ManageYear(i)).and.(IDoy + 1 == ManageDay(i))) then
            if (ManageType(i) == 'sowing' .or. ManageType(i) == &
               'planting') then
               call cmdelvar ('Plantingdate')
               call cmdelvar ('npl')
               if (manageinamount(i) < 10000.) then
                  write(*,5000) ManagementDataFile, manageinamount(i)
5000 format(' in management file ', a80, /, &
       ' nr of plants was ', f15.5)
                  stop
               end if
               call putsrp (xMod, 'npl', manageinamount(i))
               call putsi  (xMod, 'plantingdate', manageday(i))
            elseif ((ManageType(i) == 'irrigation').and. &
               ((DoWaterStress) .or. DoNitroStress)) then
               Irrigation = ManageInAmount(i)
               if (ICMODE) then
                  call getsrtm (xMod,'Irrigation',IrrigationICASA,&
                     -99.)
               end if
            elseif ((ManageType(i) == 'fertilization').and. &
               ((DoWaterStress) .or. DoNitroStress)) then
               if (fertilizerdatafile .eq. Emptyfile) then
                  if (ICMODE) then
                     call fatalerr (xMod, 'Please define the &
                        fertilizerdatafile in the PTS-file')
                  else
                     call fatalerr (xMod, 'Please define the &
                        fertilizerdatafile in the COMBCTL-file')
                  end if
               endif
               open (Unit = TempUnit, File = FertilizerDataFile, &
                  status = 'old')
               read (TempUnit, '(A)') HeaderString
3              read (TempUnit, *, end = 4) FertName, &
                  (FertChar(ii), ii=1,NrFertChar)
               call Lowerc(FertName)
               if (trim(adjustl(FertName)) == &
                  trim(adjustl(ManageInType(i)))) then
                  do ii = 1, NrFertChar
                     if (ii /= MinNIndex) then
                        if(FertChar(CIndex) <= 1.e-5) then
!                                anorganic form of nutrients
                           if (ii == DMIndex) then
                              totnan(ii) = totnan(ii)+ FertChar(ii) * &
                                 ManageInAmount(i)
                           else
                              if (ii /= NIndex) then
                                 totnan(ii) = totnan(ii)+FertChar(ii) &
                                  * FertChar(DMIndex)*ManageInAmount(i)
                              else
                                 totnan(ii) = totnan(ii)+FertChar(ii) &
```

```
                                  * FertChar(DMIndex)*ManageInAmount(i)
                               end if
                            end if
                           else
!                                organic form of nutrients
                            if (ii == DMIndex) then
                               totnor(ii) = totnor(ii) + FertChar(ii)* &
                                  ManageInAmount(i)
                            else
                               if (ii == NIndex) then
                                  totnor(ii) =totnor(ii)+FertChar(ii) *&
                                  FertChar(DMIndex) * (1. - FrMinNOrg *&
                                  FertChar(MinNIndex) * 0.1) *
                                  ManageInAmount(i)
                               else
                                  totnor(ii) = totnor(ii)+FertChar(ii)&
                                  * FertChar(DMIndex)*ManageInAmount(i)
                                  totnan(ii) =totnan(ii)+FertChar(ii)*&
                                  FertChar(DMIndex)*FertChar(MinNIndex)&
                                  * FrMinNOrg * 0.1 * ManageInAmount(i)
                               end if
                            end if
                         end if
                      end if
                   end do
                   if (FertChar(CIndex) /= 0) then
                      watermanure = watermanure + ManageInAmount(i) -&
                         totnor(DMIndex)
                   end if
                   close (TempUnit)
                   fertdepth = managedepth(i)
                   goto 5
                end if
                goto 3
4               headerstring = 'Not found: Fertilizer type ' // &
                   ManageInType(i)
                call fatalerr (xMod, headerstring)
             elseif ((ManageType(i)=='fumigation').and.(DoBlightStress&
               .or. Dosoilblight)) then
                if (pesticidedatafile .eq.  Emptyfile) then
                   if (ICMODE) then
                      call fatalerr (xMod, 'Please define the &
                         pesticidedatafile in the PTS-file')
                   else
                      call fatalerr (xMod, 'Please define the &
                         pesticide datafile in the COMBCTL-file')
                   end if
                endif
                ipesticide = ipesticide + 1
                PesticideNames(Ipesticide) = &
                   trim(adjustl(ManageInType(i)))
                call lowerc(PesticideNames(Ipesticide))
                PesticideAmount(IPesticide) = ManageInAmount(i)
                if (NrPesticides > 0) then
                   do ii = 1, NrPesticides
                      if (trim(adjustl(PesticidesNamesUsed(ii))) == &
                         PesticideNames(Ipesticide)) then
                         PesticideAmountCum(ii) =PesticideAmountCum(ii)&
                            + PesticideAmount(IPesticide)
                         goto 9
                      end if
                   end do
                end if
                NrPesticides = NrPesticides + 1
                PesticidesNamesUsed(NrPesticides) = &
                   PesticideNames(Ipesticide)
                PesticideAmountCum(NrPesticides) = &
                   PesticideAmount(IPesticide)
```

```
9             continue
            else if (ManageType(i) == 'killcrop_mechanically') then
               continue
            else if (ManageType(i) == 'killcrop_chemically') then
               continue
            else if (ManageType(i) == 'harvesting') then
               continue
            else if (ManageType(i) == 'harrowing') then
                pldep = managedepth(i)
                if (pldep <= 0.) then
                   pldep = 0.1
                end if
            else if ((Managetype(i) == 'plowing')  .or. &
                     (Managetype(i) == 'ploughing') ) then
                pldep = managedepth(i)
                if (pldep <= 0.) then
                   pldep = 0.2
                end if
                soilactivity = 'ploughing'
            else if ((Managetype(i) == 'ridging') .or. &
                      (Managetype(i) == 'hilling')) then
                soilactivity = 'soil_hilling'
            else
               call fatalerr (xMod, 'Unknown managementtype ' // &
                   Managetype(i))
                continue
            end if
5           ManageDone(i) = .true.
         end if
      end if
   end do
else if (xNewTask == 'export_states') then
   if ((ICMODE).and. ((DoWaterStress) .or. DoNitroStress)) then
      if(IrrigationICASA <= -98.999) then
         call cmdelvar ('irrigation')
         call putsrt (xMod, 'Irrigation', Irrigation)
      elseif((abs(IrrigationICASA-Irrigation) > 1.e-3) .and. &
         (irrigation >= 1.e-6)  .and. (irrigationicasa >= 1.e-6)) then
         call fatalerr (xMod, 'Irrigation in ICASA files differs &
            from irrigation in managment files')
      elseif(irrigation > 1.e-6) then
         call cmdelvar ('irrigation')
         call putsrt (xMod, 'Irrigation', Irrigation)
      else
         call cmdelvar ('irrigation')
         call putsrt (xMod, 'Irrigation', IrrigationICASA)
      end if
   elseif((irrigation > 1.e-6).and. ((DoWaterStress) .or. &
      DoNitroStress)) then
      call cmdelvar ('irrigation')
      call putsrt (xMod, 'Irrigation', Irrigation)
   else
      call cmdelvar ('irrigation')
   end if
   call cmdelvar ('pldep')
   call putsrt (xMod, 'pldep', pldep)
   if (DoNitroStress) then
      call cmdelvar ('qafer')
      call putsrt (xMod, 'QAFER', TotNan(NIndex))
      call cmdelvar ('qofer')
      call putsrt (xMod, 'QOFER', TotNor(DMIndex))
      call cmdelvar ('cnfert')
      call putsrt (xMod, 'CNFERT', TotNor(CIndex)/max(1.e-10, &
         TotNor(NIndex)))
      call cmdelvar ('cfert')
      call putsrt (xMod, 'CFERT', TotNor(CIndex)/max(1.e-10, &
         TotNor(DMIndex)))
      call cmdelvar ('NOrgFert')
```

```
      call putsrt (xMod, 'NOrgFert', TotNor(NIndex))
      call cmdelvar ('fertdepth')
      call putsrt (xMod, 'FertDepth', FertDepth)
   end if
   if ((DoNitroStress) .or. (DoWaterStress)) then
      call cmdelvar ('watermanure')
      call putsrt(xMod, 'WaterManure',  WaterManure*1.e-4)
   end if
   if (DoBlightStress .or. Dosoilblight) then
      call putsi (xMod, 'IPesticide', IPesticide)
      if (IPesticide /= 0) then
         open (Unit = TempUnit, File = PesticideDataFile, status='old')
         do ip = 1, IPesticide
            rewind (TempUnit)
            read (TempUnit, *) HeaderString
7           read (TempUnit, *, end = 6) PestName,(PestChar(iii),&
               iii=1,NrPestChar)
            call Lowerc(PestName)
            if (trim(adjustl(PestName)) == &
               trim(adjustl(PesticideNames(ip)))) then
               do iii = 1, NrPestChar
                  PestCharUsed(iii, ip) = PestChar(iii)
               end do
               goto 8
            end if
            goto 7
6           headerstring = 'Not found in file ' // PesticideDataFile//&
               ' : Pesticide type ' // PesticideNames(ip)
            call fatalerr (xMod, headerstring)
8           continue
         end do
         close (TempUnit)
         call putac (xMod, 'PesticideNames',PesticideNames, &
            IPesticide)
         call putart (xMod, 'PesticidesAmounts', PesticideAmount, &
            IPesticide)
         do iii = 1, NrPestChar
            do ip = 1, IPesticide
               PestChars(ip) = PestCharUsed(iii,ip)
            end do
            call putart (xMod, trim(adjustl(PestCharNames(iii))), &
               PestChars, IPesticide)
         end do
      end if
   end if
   if ((DoWaterStress) .or. (DoNitroStress)) then
      call getnewsoilprofile (xNewTask, 'GetNewSoilProfile', &
         xLogFileUnit, soilactivity, pldep)
   end if
end if
end subroutine Management
```

I.22. Subroutine GETNEWSOILPROFILE

```
subroutine getnewsoilprofile (xNewTask,xMod,xLogFileUnit,&
   soilactivity, pldep)
implicit none
include '..\GetSoilData\SoilDataSet.inc'
character*(*) :: xNewTask,xMod
character*30 :: soilactivity
integer :: xLogFileUnit, i, ii, ilast
real :: rugheightbottom, betweenrugdistance, pldep, newvolumehill, &
   oldvolumehill, depthmax, exposedtot, depthlayer, usedvolume,&
   RelMassFlow, pexposedl, CLOMAv, CSOMAv, CDPMAv, CSPMAv, CRPMAv, &
   ANLAYAv, FSOMAv, NSOMAv, QAFERonSoilAv, watinlayer, PerCX, PerNX, &
   SoilTpX,WatLayerX,CLOMX,CSOMX, CDPMX, CSPMX, CRPMX, ANLAYX, NSOMX, &
   FSOMX, QAFERonSoilX, frvoltot, sumvol, zeqti, thkl, AbsMassFlow, &
   totwatmovedorg , X, totwatmovednew = 0.
real, dimension(nlmxn) :: widthlayer, volumelayer, watlayer, fsom
if (nl <= 0) return
if (xNewTask == 'export_states') then
   call lowerc(soilactivity)
100 if (istart < nl+1) then
       if(soilactivity == 'soil_hilling') then
          call getsrpm (xMod, 'pexposedl',      pexposedl,    1.)
          call getsrpm (xMod, 'rugheightbottom',rugheightbottom, -99.)
          call getsrpm(xMod,'betweenrugdistance',betweenrugdistance,
             -99.)
          call getarpm (xMod, 'KST'   ,KST   ,NlMxn,i, -99.)
          call getarpm (xMod, 'WCST'  ,WCST  ,NlMxn,i, -99.)
          call getarpm (xMod, 'VGA'   ,VGA   ,NlMxn,i, -99.)
          call getarpm (xMod, 'VGL'   ,VGL   ,NlMxn,i, -99.)
          call getarpm (xMod, 'VGR'   ,VGR   ,NlMXn,i, -99.)
          call getarpm (xMod, 'VGN'   ,VGN   ,NlMXn,i, -99.)
          if((rugheightbottom<0.) .or. (betweenrugdistance < 0.) ) then
             call fatalerr (xMod, 'no value given for rugheightbottom &
                and/or betweenrugdistance ')
          else
             newvolumehill = 0.5  * 0.5 * betweenrugdistance * &
                rugheightbottom
             oldvolumehill = 0.5 * 2.* rugheightbottom * &
                betweenrugdistance - newvolumehill
          end if
         if(tkl(2) <= 1.e-8) then
            istart = 1
            tkl(2) = rugheightbottom / 2.
            tkl(1) = rugheightbottom / 2.
            ULimit(1) = - rugheightbottom
            LLimit(1) = ULimit(1) + TKL(1)
            ULimit(2) = LLimit(1)
            LLimit(2) = ULimit(2) + TKL(2)
            widthlayer(2) = 0.5 * betweenrugdistance
            VolumeLayer(1) = 0.5 * ( 0.5* 0.5*betweenrugdistance + 0.)&
               * tkl(1) * 1.
            VolumeLayer(2) = 0.5 * ( 0.5*betweenrugdistance + 0.5 * &
               0.5*betweenrugdistance) * tkl(2) * 1.
            frVolumeLayer(1) = VolumeLayer(1) / (betweenrugdistance * &
               tkl(1) * 1.)
            frVolumeLayer(2) = VolumeLayer(2) / (betweenrugdistance * &
               tkl(2) * 1.)
            exposedlayer(1)  = 2. * sqrt(((0.5**3) * &
               betweenrugdistance)**2 + tkl(1)**2 )
            exposedlayer(2)  = exposedlayer(1)
            if (exposedlayer(1) > 0.) then
               exposedlayer(1) = exposedlayer(1)**pexposedl
            end if
            if (exposedlayer(2) > 0.) then
               exposedlayer(2) = exposedlayer(2)**pexposedl
            end if
```

```
66                depthmax = 0.
67                exposedtot = exposedlayer(1) + exposedlayer(2)
68                call getsrp (xMod, 'ZEQTI ',ZEQTI )
69                do ii = 1,2
70                   ZEQT(i)   = ZEQTI
71                   WCST(ii)  = WCST(3)
72                   KST(ii)   = kst(3)
73                   VGA(ii)   = VGA(3)
74                   VGN(ii)   = VGN(3)
75                   VGL(ii)   = VGL(3)
76                   VGR(II)   = VGR(3)
77                   RHOD(ii)  = 0.
78                   PerC(ii)  = 0.
79                   PerN(ii)  = 0.
80                   SoilTp(ii)= 0.
81                   WatLayer(ii) = 0.
82                   CLOM(ii)  = 0.
83                   CSOM(ii)  = 0.
84                   CDPM(ii)  = 0.
85                   CSPM(ii)  = 0.
86                   CRPM(ii)  = 0.
87                   ANLAY(ii) = 0.
88                   NSOM(ii)  = 0.
89                   FSOM(ii)  = 0.
90                   QAFERonSoil(ii) = 0.
91                end do
92                totwatmovedorg = 0.
93                totwatmovednew = 0.
94                do i = 3, Nl
95                   if (rugheightbottom - ulimit(i) > 1.e-6) then
96                      ZEQT(i)          = ZEQTI
97                      thkl             = min (rugheightbottom, llimit(i))
98                      depthlayer       = rugheightbottom + thkl
99                      thkl             = max(0., thkl - ulimit(i))
100                     widthlayer(i)    = depthlayer *  betweenrugdistance/&
101                        (2. * rugheightbottom)
102                     volumelayer(i)   = 0.5 * thkl * (widthlayer(i) + &
103                        widthlayer(i-1))
104                     usedvolume       = max(1.e-10, betweenrugdistance * &
105                        tkl(i) - volumelayer(i))
106                     frvolumelayer(i) = volumelayer(i) / max(1.e-10, &
107                        betweenrugdistance * tkl(i))
108                     exposedlayer(i)  = 2. * sqrt(0.5*(widthlayer(i)- &
109                        widthlayer(i-1))**2 + thkl**2)
110                     if (exposedlayer(i) > 0.) then
111                        exposedlayer(i) = exposedlayer(i)**pexposedl
112                     end if
113                     exposedtot = exposedtot + exposedlayer(i)
114                     CLOMAv  = clom(i)
115                     CSOMAv  = csom(i)
116                     CDPMAv  = cdpm(i)
117                     CSPMAv  = cspm(i)
118                     CRPMAv  = crpm(i)
119                     ANLAYAv = anlay(i)
120                     NSOMAv  = nsom(i)
121                     QAFERonSoilAv = QAFERonSoil(i)
122                     watinlayer = tkl(i) * 1000. * WCLQT(i)
123                     totwatmovedorg = totwatmovedorg + watinlayer
124                     watlayer(i) = watinlayer
125                     FSOMAv  = FSOM(i)
126                     RelMassFlow = usedvolume / NewVolumeHill
127                     AbsMassFlow = usedvolume / (volumelayer(i) + &
128                        usedvolume)
129                     do ii = 1,2
130                        X = AbsMassFlow * VolumeLayer(ii) / NewVolumeHill
131                        RHOD(ii)  = RHOD(ii)  + RHOD(i)   * RelMassFlow
132                        PerC(ii)  = PerC(ii)  + PerC(i)   * RelMassFlow
133                        PerN(ii)  = PerN(ii)  + PerN(i)   * RelMassFlow
```

```
                    SoilTp(ii)= SoilTp(ii) + SoilTp(i) * RelMassFlow
                    WatLayer(ii) = WatLayer(ii) + watinlayer * &
                       X / NewVolumeHill
                    WatLayer(i)  = WatLayer(i)  - watinlayer * &
                       X / NewVolumeHill

                    CLOM(ii)  = CLOM(ii)  + clomav   * X
                    CSOM(ii)  = CSOM(ii)  + csomav   * X
                    CDPM(ii)  = CDPM(ii)  + cdpmav   * X
                    CSPM(ii)  = CSPM(ii)  + cspmav   * X
                    CRPM(ii)  = CRPM(ii)  + crpmav   * X
                    ANLAY(ii) = ANLAY(ii) + anlayav  * X
                    NSOM(ii)  = NSOM(ii)  + nsomav   * X
                    FSOM(ii)  = FSOM(ii)  + Fsomav   * X
                    QAFERonSoil(ii) = QAFERonSoil(ii)+QAFERonSoilAv*X
                    CLOM(i)   = CLOM(i)   - clomav   * X
                    CSOM(i)   = CSOM(i)   - csomav   * X
                    CDPM(i)   = CDPM(i)   - cdpmav   * X
                    CSPM(i)   = CSPM(i)   - cspmav   * X
                    CRPM(i)   = CRPM(i)   - crpmav   * X
                    ANLAY(i)  = ANLAY(i)  - anlayav  * X
                    NSOM(i)   = NSOM(i)   - nsomav   * X
                    FSOM(i)   = FSOM(i)   - fsomav   * X
                    QAFERonSoil(i) = QAFERonSoil(i) - QAFERonSoilAv*X
                 end do
              end if
           end do
           do i = 1, nl
              frexposedlayer(i)=exposedlayer(i)/max(1.e-10,exposedtot)
              if(vgn(i) < 1. + tiny2) then
                 vgn(i) = 1. + tiny2
              end if
              if(wcst(i) > 1.-tiny2) then
                 wcst(i) = 1.-tiny2
              endif
              if(wcst(i) < 0.05) then
                 wcst(i) = 0.05
              endif
              if (rugheightbottom - ulimit(i) > 1.e-6) then
                 totwatmovednew = totwatmovednew + watlayer(i)
                 wclqt(i)=WatLayer(i)/(frvolumelayer(i)*tkl(i)*1000.)
                 nlexp = i
              end if
           end do
           wcst(1) = wcst(3)
           WCAD(1) = WCAD(3)
           WCFC(1) = WCFC(3)
           WCWU(1) = WCWU(3)
           WCWO(1) = WCWO(3)
           WCWP(1) = WCWP(3)
           wcst(2) = wcst(3)
           WCAD(2) = WCAD(3)
           WCFC(2) = WCFC(3)
           WCWU(2) = WCWU(3)
           WCWO(2) = WCWO(3)
           WCWP(2) = WCWP(3)
        else
           PerCX  = 0.
           PerNX  = 0.
           SoilTpX= 0.
           WatLayerX = 0.
           CLOMX  = 0.
           CSOMX  = 0.
           CDPMX  = 0.
           CSPMX  = 0.
           CRPMX  = 0.
           ANLAYX = 0.
           NSOMX  = 0.
```

```
202               FSOMX  = 0.
203               QAFERonSoilX = 0.
204               frvoltot = 0.
205               sumvol = 0.
206               do ii = 1, Nl
207                  if (rugheightbottom - ulimit(ii) <= 1.e-8) goto 505
208                  frvoltot = frvoltot + frvolumelayer(ii)
209                  PerCX  = PerCX + PerC(ii) * frvolumelayer(ii)
210                  PerNX  = PerNX + PerN(ii) * frvolumelayer(ii)
211                  SoilTpX= SoilTpX+SoilTp(ii)*frvolumelayer(ii) * tkl(ii)
212                  WatLayerX = WatLayerX + frvolumelayer(ii) * tkl(ii) * &
213                     1000. * WCLQT(ii)
214                  sumvol = sumvol + frvolumelayer(ii) * tkl(ii)
215                  CLOMX  = CLOMX + CLOM(ii)
216                  CSOMX  = CSOMX + CSOM(ii)
217                  CDPMX  = CDPMX + CDPM(ii)
218                  CSPMX  = CSPMX + CSPM(ii)
219                  CRPMX  = CRPMX + CRPM(ii)
220                  ANLAYX = ANLAYX + ANLAY(ii) + QAFERonSoil(ii)
221                  NSOMX  = NSOMX + NSOM(ii)
222                  FSOMX  = FSOMX + FSOM(ii)
223                  QAFERonSoil(ii) = 0.
224               end do
225   505         ilast = max(1, ii - 1)
226               do ii = 1, ilast
227                  PerC(ii)   = PerCX * frvolumelayer(ii) / frvoltot
228                  PerN(ii)   = PerNX * frvolumelayer(ii) / frvoltot
229                  SoilTP(ii) = SoilTpX / sumvol
230                  WCLQT(ii)  = WatLayerX / (sumvol * 1000.)
231                  CLOM(ii)   = CLOMX * frvolumelayer(ii) / frvoltot
232                  CSOM(ii)   = CSOMX * frvolumelayer(ii) / frvoltot
233                  CDPM(ii)   = CDPMX * frvolumelayer(ii) / frvoltot
234                  CSPM(ii)   = CSPMX * frvolumelayer(ii) / frvoltot
235                  CRPM(ii)   = CRPMX * frvolumelayer(ii) / frvoltot
236                  ANLAY(ii)  = ANLAYX * frvolumelayer(ii) / frvoltot
237                  NSOM(ii)   = NSOMX * frvolumelayer(ii) / frvoltot
238               end do
239            endif
240            call cmdelvar( 'ulimit')
241            call cmdelvar( 'llimit')
242            call cmdelvar( 'VGR')
243            call cmdelvar( 'WCST')
244            call cmdelvar( 'KST')
245            call cmdelvar( 'VGA')
246            call cmdelvar( 'VGN')
247            call cmdelvar( 'VGL')
248            call cmdelvar( 'wclqt')
249            call cmdelvar( 'WCFC')
250            call cmdelvar( 'WCWP')
251            call cmdelvar( 'WCAD')
252            call cmdelvar( 'WCWU')
253            call cmdelvar( 'WCWO')
254            call cmdelvar( 'RHOD')
255            call cmdelvar( 'PerC')
256            call cmdelvar( 'PerN')
257            call cmdelvar( 'CLOM')
258            call cmdelvar( 'CSOM')
259            call cmdelvar( 'CDPM')
260            call cmdelvar( 'CSPM')
261            call cmdelvar( 'CRPM')
262            call cmdelvar( 'ANLAY')
263            call cmdelvar( 'NSOM')
264            call cmdelvar( 'FSOM')
265            call cmdelvar( 'QAFERonSoil')
266            call putarp (xMod, 'ulimit', ulimit, Nl)
267            call putarp (xMod, 'llimit', llimit, Nl)
268            call putarp (xMod, 'VGR',VGR, Nl)
269            call putarp (xMod, 'WCST',WCST, Nl)
```

```
            call putarp (xMod, 'KST',KST, Nl)
            call putarp (xMod, 'VGA',VGA, Nl)
            call putarp (xMod, 'VGN',VGN, Nl)
            call putarp (xMod, 'VGL',VGL, Nl)
            call putart (xMod, 'wclqt',wclqt, Nl)
            call putarp (xMod, 'RHOD',RHOD, Nl)
            call putart (xMod, 'PerC',PerC, Nl)
            call putart (xMod, 'PerN',PerN, Nl)
            call putart (xMod, 'CLOM',CLOM, Nl)
            call putart (xMod, 'CSOM',CSOM, Nl)
            call putart (xMod, 'CDPM',CDPM, Nl)
            call putart (xMod, 'CSPM',CSPM, Nl)
            call putart (xMod, 'CRPM',CRPM, Nl)
            call putart (xMod, 'ANLAY',ANLAY, Nl)
            call putart (xMod, 'NSOM',NSOM, Nl)
            call putart (xMod, 'FSOM',FSOM, Nl)
            call putart (xMod, 'QAFERonSoil',QAFERonSoil, Nl)
            call putarp (xMod, 'WCFC',WCFC, Nl)
            call putarp (xMod, 'WCWP',WCWP, Nl)
            call putarp (xMod, 'WCAD',WCAD, Nl)
            call putarp (xMod, 'WCWU',WCWU, Nl)
            call putarp (xMod, 'WCWO',WCWO, Nl)
            do i = 1, nl
               exposedlayer(i) = exposedlayer(i) / max(1.e-10, exposedtot)
            end do
         elseif (soilactivity == 'ploughing') then
            if (pldep > 0.40) then
               call PedoTrans_VGN(xLogFileUnit, pldep)
            end if
         end if
      end if
   end if
   end subroutine getnewsoilprofile
```

I.23. Subroutine NDEM4POS

```
subroutine NDem4Pos (xNewTask,xMod,xLogFileUnit)
implicit none
include '..\GetCropData\CropDataSet.inc'
character*(*) :: xNewTask,xMod
integer :: xLogFileUnit, iyear, i,tmpi1,RootLayerNo
real :: Delt, LeafArGrowthP, TuberWtGrowthP, StemVolGrowthP, &
   LeafWtGrowthP, StemWtGrowthP, TotWtGrowthP, TotalDryWt, &
   RNStabLeafNewP, RNStabStemNewP, RNStabTuberNewP, RNStrucLeafNew,&
   RNStrucStemNew, RNStrucTuberNew, RNTuberResp, StrucDMTuberLive, &
   RelRespTuber, nuptmin, ndemrest,nsol,nstabstemlive, nstabtuberlive,&
   NstabLeafMax,NStabStemMax,NStabTuberMax, NstabLeafMin,NStabStemMin,&
   NStabTuberMin, NstrucLeafMin, NStrucStemMin, NStrucTuberMin, &
   kdestableaf, kdestabstem, kdestabtuber, RNSolNewP, StemVol, &
   StressIndexStAgP,StressIndexRoAgP,StressIndexTuAgP,&
   RNStabLeafNewMax, RNStabLeafNewMin, RNStabStemNewMax, &
   RNStabStemNewMin, RNStabTuberNewMax, RNStabTuberNewMin, &
   RNTotStabMax, RNTotStabMin, RNTotStruc, RNTotGrowthMax, &
   RNTotGrowthMin, RedistrNTot, ReserveNTot, FrNReserveUse,&
   FrNRedistrUse,FrNAvail,AllowedAboveGrowthNlim,AllowedTuberGrowthNlim
real :: RedistrLeafNDead, ReserveLeafNLive, RedistrStemNDead, &
   ReserveStemNLive, RedistrTuberNDead, ReserveTuberNLive, RedisN, &
   ReserN, NAvail, NavailGrowth, NuptMaxP, NuptMinP
logical RootGrowthFlag, RootGrowthFlagSoil
character*13 OldTask
save
if (xNewTask == 'initialize') then
   call Logmessage (xLogFileUnit,9,xMod,'3.0')
   call getsrp (xMod,'delt',Delt)
   call getsrp(xMod, 'NStabLeafMax'  ,NStabLeafMax)
   call getsrp(xMod, 'NstabStemMax'  ,NstabStemMax)
   call getsrp(xMod, 'NstabTuberMax' ,NstabTuberMax)
   call getsrp(xMod, 'NStabLeafMin'  ,NStabLeafMin)
   call getsrp(xMod, 'NstabStemMin'  ,NstabStemMin)
   call getsrp(xMod, 'NstabTuberMin' ,NstabTuberMin)
   NStabLeafMin  = min(NStabLeafMin,  NStabLeafMax)
   NStabStemMin  = min(NStabStemMin,  NStabStemMax)
   NStabTuberMin = min(NStabTuberMin, NStabTuberMax)
   call getsrp(xMod, 'NstrucLeafMin' ,NstrucLeafMin)
   call getsrp(xMod, 'NStrucStemMin' ,NStrucStemMin)
   call getsrp(xMod, 'NstrucTuberMin',NstrucTuberMin)
   call getsrp(xMod, 'ksynstab'      ,ksynstab)
   call getsrp(xMod, 'kdestableaf'   ,kdestableaf)
   call getsrp(xMod, 'kdestabstem'   ,kdestabstem)
   call getsrp(xMod, 'kdestabtuber'  ,kdestabtuber)
   call getsrp(xMod, 'RelRespTuber', RelRespTuber)
   NuptMin  = 0.
   NuptMinP = 0.
   NuptMaxP = 0.
   AllowedAboveGrowthNlim = 1.
   AllowedTuberGrowthNlim = 1.
else if (xNewTask == 'do_rates') then
   call getsi (xMod, 'LeafClCnt', LeafClCnt)
   NuptMin = 0.
   if (LeafClCnt > 0) then
      call getsrt (xMod, 'NStabStemLive'    , NStabStemLive)
      call getsrt (xMod, 'NStabTuberLive'   , NStabTuberLive)
      call getsrt (xMod, 'StressIndexStAgP' , StressIndexStAgP)
      call getsrt (xMod, 'StressIndexRoAgP' , StressIndexRoAgP)
      call getsrt (xMod, 'StressIndexTuAgP' , StressIndexTuAgP)
      call getsrt (xMod, 'LeafArGrowthP'    , LeafArGrowthP)
      call getsrt (xMod, 'StemVolGrowthP'   , StemVolGrowthP)
      call getsrt (xMod, 'LeafWtGrowthP'    , LeafWtGrowthP)
      call getsrt (xMod, 'StemWtGrowthP'    , StemWtGrowthP)
      call getsrt (xMod, 'TuberWtGrowthP'   , TuberWtGrowthP)
      call getsrt (xMod, 'TotalDryWt'       , TotalDryWt)
```

```
      call GETSRT (xMod, 'LAIEff'           , LAIEff)
      call getsrt (xMod, 'stemvol'          , stemvol)
      call getsrt (xMod, 'nsol'             , nsol)
      call getsrt (xMod, 'StrucDMTuberLive'  , StrucDMTuberLive)
      RNStabLeafNewMax  = (LeafArGrowthP/10000.) * NstabLeafMax
      RNStabStemNewMax  = StemVolGrowthP  * NStabStemMax
      RNStabTuberNewMax = TuberWtGrowthP  * NStabTuberMax
      RNStabLeafNewMin  = (LeafArGrowthP/10000.) * NstabLeafMin
      RNStabStemNewMin  = StemVolGrowthP  * NStabStemMin
      RNStabTuberNewMin = TuberWtGrowthP  * NStabTuberMin
      RNStrucLeafNew  = (LeafArGrowthP/10000.) * NstrucLeafMin
      RNStrucStemNew    = StemVolGrowthP  * NStrucStemMin
      RNStrucTuberNew   = TuberWtGrowthP  * NStrucTuberMin
      RNTotStabMax =RNStabLeafNewMax+RNStabStemNewMax+RNStabTuberNewMax
      RNTotStabMin=RNStabLeafNewMin+RNStabStemNewMin +RNStabTuberNewMin
      RNTotStruc     = RNStrucLeafNew + RNSTrucStemNew +RNStrucTuberNew
      RNTotGrowthMax = RNTotStabMax + RNTotStruc
      RNTotGrowthMin = RNTotStabMin + RNTotStruc
      RNTuberResp = NStabTuberLive * max(0., StrucDMTuberLive * &
         RelRespTuber ) / max(1.e-10, TuberWt)
      RedistrLeafNDead = 0.
      ReserveLeafNLive = 0.
      do i = 1, leafclcnt
         if ( (not(LeafAliveCl(i))) .and. (NStabLiveCl(i) > 0.)) then
            RedistrLeafNDead    = RedistrLeafNDead + NStabLiveCl(i)
         elseif (LeafAliveCl(i)) then
            ReserveLeafNLive    = ReserveLeafNLive + max(0., &
               (NStabLiveCl(i) - NStabLeafMin * LeafArLiveCl(i) / &
               10000.)) * kdestabLeaf
         end if
      end do
      ReserveStemNLive  = (1. - StressIndexStAgP) * max(0., &
         (NStabStemLive  - NStabStemMin  * StemVol ) ) * kdestabStem
      ReserveTuberNLive = (1. - StressIndexTuAgP) * max(0., &
         (NStabTuberLive - NStabTuberMin * TuberWt - RNTuberResp) ) * &
         kdestabTuber
      RedistrStemNDead  = StressIndexStAgP * NStabStemLive
      RedistrTuberNDead = StressIndexTuAgP * NStabTuberLive
      RedistrNTot =RedistrLeafNDead+RedistrStemNDead+RedistrTuberNDead
      ReserveNTot = ReserveLeafNLive+ReserveStemNLive+ReserveTuberNLive
      NAvailGrowth = ksynstab * NSol + RedistrNTot + ReserveNTot + &
         RNTuberResp
      AllowedAboveGrowthNlim = 1.
      AllowedTuberGrowthNlim = 1.
      if (NAvailGrowth < RNTotGrowthMin .and. RNTotGrowthMin > 1.e-8) &
         then
         NDemRest = RNStabLeafNewMin + RNStabStemNewMin +  &
                    RNStrucLeafNew   + RNStrucStemNew
         if (NDemRest > 1.e-8) then
            if (NAvailGrowth <= NDemRest ) then
               AllowedAboveGrowthNlim = NAvailGrowth / NDemRest
               AllowedTuberGrowthNlim = 0.
            else
               AllowedAboveGrowthNlim = 1.
               AllowedTuberGrowthNlim = max(0., NAvailGrowth - &
                  NdemRest)/max(1.e-8, RNStabTuberNewMin + &
                  RNStrucTuberNew)
            end if
         else
            AllowedTuberGrowthNlim = max(0.,NAvailGrowth / max(1.e-8, &
               RNTotGrowthMin))
         endif
      elseif (RNTotGrowthMin <= 1.e-8) then
         AllowedAboveGrowthNlim = 0.
         AllowedTuberGrowthNlim = 0.
      end if
      NuptMaxP = max(0., RNTotGrowthMax)
      NuptMinP = max(0., RNTotGrowthMin)
```

```
134       end if
135       call putsrt (xMod, 'AllowedAboveGrowthNLim', AllowedAboveGrowthNlim)
136       call putsrt (xMod, 'AllowedTuberGrowthNLim', AllowedTuberGrowthNlim)
137       call putsrt (xMod, 'NuptMaxP', NuptMaxP)
138       call putsrt (xMod, 'NuptMinP', NuptMinP)
139    else if (xNewTask == 'output') then
140       continue
141    else if (xNewTask == 'do_states') then
142       continue
143    else if (xNewTask == 'export_states') then
144       continue
145    else if (xNewTask == 'terminate') then
146       continue
147    end if
148    return
149    end subroutine NDem4Pos
```

I.24. Subroutine NDEM4REA

```
subroutine NDem4Rea (xNewTask,xMod,xLogFileUnit)
implicit none
include '..\GetCropData\CropDataSet.inc'
character*(*) :: xNewTask,xMod
character*10 :: FINTtext
character*30 :: namerates(8), namestates(32)
integer :: xLogFileUnit, iyear, datecmp, datecmp2, datecmp3, &
   leafclstart, leafclend, leafclcntlast, lastleaf, lastleafold, &
   integer i,tmpi1,RootLayerNo, FintExt
real :: Delt, LeafArGrowth, LeafWtGrowth, StemWtGrowth, TuberWtGrowth,&
   TotWtGrowth, StemVolGrowth, nuptr, NDilutionPar, nsol, rnsol, &
   nsolconc, NSolDead, RNSolDead, NSolLoss, RNSolLoss, nstableaflive, &
   rnstableaflive, nstabconcleaf, nstabconcleaflive,nstabconcleafdead,&
   nstabconcleafloss, nstabconcleafwt, nstrucleaflive, nstrucleafwt, &
   rnstrucleaflive, nstrucconcleaf, nstrucconcleafwt, ntotleaflive, &
   ntotconcleaf, ntotconcleafwt, ntotstemlive, ntotconcstem, &
   ntottuberlive, ntotconctuber, ntottotallive, ntotconctotal, &
   ntotabovelive, ntotconcabove, ntotunderlive, ntotconcunder, &
   NstabconcLeafPhot,ntotconcleafwtphot,ntottuberloss,ntotconcabovewt,&
   Ntotconcabovear, nstabstemlive, rnstabstemlive, rnstabstemdead,&
   rnstabstemloss, nstabconcstem, nstabconcstemvol, nstrucstemlive,&
   rnstrucstemlive, rnstrucstemdead, rnstrucstemloss, nstrucconcstem,&
   nstrucrootlive, rnstrucrootlive, nstrucconcroot, nstabtuberlive, &
   rnstabtuberlive, nstabconctuber, nstructuberlive, rnstructuberlive,&
   nstrucconctuber, NTotTuberDead, nstableafdead, nstableafloss, &
   rnstableafdead, rnstableafloss, nstrucleafdead, nstrucleafloss, &
   rnstrucleafdead, rnstrucleafloss, nstrucconcleafdead, sumtrans, &
   leafage, NStabLiveClnew, StemVol, NstabLeafMax, NStabStemMax, &
   NStabTuberMax, NstabLeafMin, NStabStemMin, NStabTuberMin, &
   NstrucLeafMin, NStrucStemMin, NStrucTuberMin, RNStabLeafNew, &
   RNStabStemNew, RNStabTuberNew, RNStrucLeafNew, RNStrucStemNew, &
   RNStrucTuberNew, kdestableaf, kdestabstem, kdestabtuber,RNshufLeaf,&
   RNshufStem, RNshufTuber, RNExtraLeaf, RNExtraStem, RNExtraTuber, &
   RNSolNew,LaiSpad, NSpadConcLeafWt, NStabMinLeaf, NStabTotLeaf, &
   NStabMinStem, NStabMinTuber, NUptakeTot, NLostLeaf, NLostStem, &
   NLostTuber, NLostAbove, NLostUnder, NinLeafI, NinStemI, NinTuberI, &
   NtotIn, NtotAccounted, NBalance, RNTotGrowth,NAvailGrowth, &
   RNshufMax,FractionShuf,DMNStabLeafMin,DMNStrucLeaf,FreeDMLeafLive, &
   DMNStabStemMin, DMNStrucStem, FreeDMStemLive, DMNStabTuberMin, &
   DMNStrucTuber, FreeDMTuberLive, NStabStemDead, NStabStemLoss, &
   NStrucStemDead, NStrucStemLoss, NtotLeafDead, NtotLeafLoss, &
   NtotStemDead, NtotStemLoss, NtotTotalDead, NtotTotalLoss, &
   NtotUnderDead, NtotAboveDead, NtotAboveLoss, NtotUnderLoss, &
   RNStabLeafNewMax,RNStabLeafNewMin,RNStabStemNewMax, &
   RNStabStemNewMin, RNStabTuberNewMax, RNStabTuberNewMin, &
   NStabTuberDead, NStrucTuberDead, NStabTuberLoss, NStrucTuberLoss, &
   RNStabTuberDead,RNStrucTuberDead, RNStabTuberLoss,RNStrucTuberLoss,&
   RNSolDyingLeaves, NStabAdd, NStabNew, SumNStab, NStabLeafPot, &
   RedistrNTot, ReserveNTot, FrNReserveUse, FrNRedistrUse, FrNAvail, &
   NBalGr, StressIndexStAg, StressIndexRoAg, StressIndexTuAg, &
   StressIndexStAgP, StressIndexRoAgP, StressIndexTuAgP, RNTotStabMax,&
   RNTotStabMin, RNTotStruc, RNTotGrowthMax, RNTotGrowthMin, SPADcalc,&
   RedistrLeafNDead,ReserveLeafNLive,RedistrStemNDead, &
   ReserveStemNLive, RedistrTuberNDead, ReserveTuberNLive, RedisN, &
   ReserN, NAvail, FreeDmLeaf, NStabLeafAvail, FrReserveUse, &
   FrRedistrUse, ReserveLeafLive, ReserveStemLive, ReserveTuberLive, &
   LaiLoss, fint, spad1, spad2, alphaNper, alphaNopn, etaSPAD, &
   kappaSPAD, RNTuberResp, StrucDMTuberLive, RelRespTuber, kspad, &p
   bspad, cspad, ProtTCalc, alphaProtT, betaProtT, ProtRCalc, &
   alphaProtR, betaProtR, NOrgConcLeafWt, NOrgConcStem, NOrgConcTuber,&
   NOrgConcAboveWt, NitrConcLeafWt, NitrConcStem, NitrConcTuber, &
   NitrConcAboveWt, NNitrLeaf, NNitrStem, NNitrTuber, NNitrAboveWt, &
   NNitrTotal, NNOrgLeaf, NNOrgStem, NNOrgTuber, NNOrgAboveWt, &
   NNOrgTotal, TotalDryWt, TotalDryWtPresent, intgrl,limit, FuncECPDF
real, dimension(leafclmxn) :: rNStabLiveCl, RNStabDeadCl,RNStabLossCl,&
```

```
      rNStrucLiveCl, RNStrucDeadCl, RNSTrucLossCl, NStrucLiveClconc, &
      partransfraction, partrans, RDMLeafCl, NStabOld
   save
   data namerates / 'LeafArGrowth', 'StemVolGrowth', 'TuberWtGrowth', &
      'StressindexStAg', 'StressIndexStAgP', 'StressIndexTuAg', &
      'StressIndexTuAgP', 'Nuptr'/
   data namestates / 'NStabLeafLive', 'NStabStemLive', 'NStabTuberLive',&
      'NStrucLeafLive','NStrucStemLive','NStrucTuberLive','NTotLeafLive',&
      'NTotStemLive', 'NTotTuberLive', 'NTotTotalLive', 'NTotAboveLive', &
      'NTotUnderLive', 'NTotConcLeaf', 'NTotConcLeafWt', 'NTotConcStem', &
     'NTotConcTuber','NTotConcAboveAr','NTotConcAboveWt','NTotConcUnder',&
      'NTotConcTotal', 'nsol', 'nsolconc', 'nstabconcleaf', &
      'nstabconcleafLive', 'nstabconcleafwt', 'nstabconcstem', &
      'nstabconcstemVol','nstabconctuber', 'nstabconcleafphot', &
      'ntotconcleafwtphot','FreeDMStemLive''FreeDMTuberLive' /
   if (xNewTask == 'initialize') then
      call Logmessage (xLogFileUnit,9,xMod,'3.0')
      call getsrp (xMod,'delt',Delt)
      NUptR           = 0.
      NstabLeafLive   = 0.
      NstabStemLive   = 0.
      NstabTuberLive  = 0.
      NstabLeafDead   = 0.
      NstabStemDead   = 0.
      NstabTuberDead  = 0.
      NstabLeafLoss   = 0.
      NstabStemLoss   = 0.
      NstabTuberLoss  = 0.
      NstrucLeafLive  = 0.
      NstrucStemLive  = 0.
      NstrucTuberLive = 0.
      NstrucLeafDead  = 0.
      NstrucStemDead  = 0.
      NStrucTuberDead = 0.
      NstrucLeafLoss  = 0.
      NstrucStemLoss  = 0.
      NStrucTuberLoss = 0.
      NSol            = 0.
      NSolDead        = 0.
      NSolLoss        = 0.
      NUptakeTot      = 0.
      NtotAboveLive   = 0.
      NtotAboveDead   = 0.
      NtotUnderLive   = 0.
      NtotUnderDead   = 0.
      NtotLeafLive    = 0.
      NtotTotalLive   = 0.
      NtotStemLive    = 0.
      NtotTuberLive   = 0.
      NtotLeafDead    = 0.
      NtotTuberDead   = 0.
      NtotTotalDead   = 0.
      NtotStemDead    = 0.
      NtotLeafLoss    = 0.
      NtotTotalLoss   = 0.
      NtotStemLoss    = 0.
      NinLeafI        = 0.
      NinStemI        = 0.
      NinTuberI       = 0.
      NtotconcLeaf    = 0.
      NtotconcLeafWt  = 0.
      NtotconcStem    = 0.
      NtotconcTuber   = 0.
      NtotconcAboveWt = 0.
      NtotconcAboveAr = 0.
      NtotconcUnder   = 0.
      NtotconcTotal   = 0.
      NstabconcLeaf   = 0.
```

```
      NstrucconcLeaf  = 0.
      NstabconcLeafWt = 0.
      NstrucconcLeafWt= 0.
      NstabconcStem   = 0.
      NstrucconcStem  = 0.
      NstabconcTuber  = 0.
      NstrucconcTuber = 0.
      nsolconc        = 0.
      NstabconcLeafPhot = 0.
      NTotconcLeafWtPhot = 0.
      NTotUnderDead   = 0.
      NtotUnderLoss   = 0.
      NOrgConcLeafWt = 0.
      NOrgConcStem = 0.
      NOrgConcTuber = 0.
      NOrgConcAboveWt = 0.
      NitrConcLeafWt = 0.
      NitrConcStem = 0.
      NitrConcTuber = 0.
      NitrConcAboveWt = 0.
      NNitrLeaf = 0.
      NNitrStem = 0.
      NNitrTuber = 0.
      NNitrAboveWt = 0.
      NNitrTotal = 0.
      NNOrgLeaf = 0.
      NNOrgStem = 0.
      NNOrgTuber = 0.
      NNOrgAboveWt = 0.
      NNOrgTotal = 0.
      SPADcalc        = 0.
      fint            = 0.
      NstabconcStemVol= 0.
      ProtTcalc       = 0.
      ProtRcalc       = 0.
      do i = 1, leafclmxn
         NStrucLiveCl(i)  = 0.
         NStabLiveCl(i)   = 0.
         rNStrucLiveCl(i) = 0.
         rNStabLiveCl(i)  = 0.
         NStrucDeadCl(i)  = 0.
         NStabDeadCl(i)   = 0.
         rNStrucDeadCl(i) = 0.
         rNStabDeadCl(i)  = 0.
         NStrucLossCl(i)  = 0.
         NStabLossCl(i)   = 0.
         rNStrucLossCl(i) = 0.
         rNStabLossCl(i)  = 0.
         FreeDMLeafCl(i)  = 0.
      end do
      call getsi (xMod, 'FintExt' , FintExt)
      call getsrp(xMod, 'NSolConcI', NSolConcI)
      call getsrp(xMod, 'NStabLeafMax',NStabLeafMax)
      call getsrp(xMod, 'NstabStemMax',NstabStemMax)
      call getsrp(xMod, 'NstabTuberMax',NstabTuberMax)
      call getsrp(xMod, 'NStabLeafMin',NStabLeafMin)
      call getsrp(xMod, 'NstabStemMin',NstabStemMin)
      call getsrp(xMod, 'NstabTuberMin',NstabTuberMin)
      NStabLeafMin  = min(NStabLeafMin,  NStabLeafMax)
      NStabStemMin  = min(NStabStemMin,  NStabStemMax)
      NStabTuberMin = min(NStabTuberMin, NStabTuberMax)
      call getsrp(xMod, 'NstrucLeafMin',NstrucLeafMin)
      call getsrp(xMod, 'NStrucStemMin',NStrucStemMin)
      call getsrp(xMod, 'NstrucTuberMin',NstrucTuberMin)
      call getsrp(xMod, 'ksynstab',ksynstab)
      call getsrp(xMod, 'kdestableaf',kdestableaf)
      call getsrp(xMod, 'kdestabstem',kdestabstem)
      call getsrp(xMod, 'kdestabtuber',kdestabtuber)
```

```
      call getsrp(xMod, 'RelLossDeadLeaves', RelLossDeadLeaves)
      call getsrp(xMod, 'RelLossDeadStems', RelLossDeadStems)
      call getsrpm(xMod, 'RelLossDeadRoots', RelLossDeadRoots, 0.)
      call getsrp(xMod, 'RelLossDeadTubers', RelLossDeadTubers)
      RelLossDeadLeaves = max(0., min(1., RelLossDeadLeaves))
      RelLossDeadStems  = max(0., min(1., RelLossDeadStems))
      RelLossDeadTubers = max(0., min(1., RelLossDeadTubers))
      call getsrp(xMod, 'NdilutionPar', NDilutionPar)
      call getsrp(xMod, 'DMNStabLeafMin',  DMNStabLeafMin)
      call getsrp(xMod, 'DMNStabSTemMin',  DMNStabStemMin)
      call getsrp(xMod, 'DMNStabTuberMin', DMNStabTuberMin)
      call getsrp(xMod, 'DMNStrucLeaf',  DMNStrucLeaf)
      call getsrp(xMod, 'DMNStrucStem',  DMNStrucStem)
      call getsrp(xMod, 'DMNStrucTuber', DMNStrucTuber)
      call getsrp(xMod, 'MaxSpad',MaxSpad)
      call getsrp(xMod, 'alphaSpad',alphaSpad)
      call getsrp(xMod, 'betaSpad',betaSpad)
      call getsrp(xMod, 'gammaSpad',gammaSpad)
      call getsrp(xMod, 'deltaSpad',deltaSpad)
      call getsrp(xMod, 'etaSpad',etaSpad)
      call getsrp(xMod, 'kappaSpad',kappaSpad)
      call getsrp(xMod, 'cSpad',cSpad)
      call getsrp(xMod, 'bSpad',bSpad)
      call getsrp(xMod, 'kSpad',kSpad)
      call getsrp(xMod, 'alphaProtR',alphaProtR)
      call getsrp(xMod, 'betaProtR',betaProtR)
      call getsrp(xMod, 'alphaProtT',alphaProtT)
      call getsrp(xMod, 'betaProtT',betaProtT)
      call getsrp(xMod, 'RelRespTuber', RelRespTuber)
      call getsrp (xMod, 'ecpdf', ecpdf)
      call getsrp (xMod, 'relLIntleafdead', relLIntleafdead)
      RNStabLeafNew    = 0.
      RNStrucLeafNew   = 0.
      RNStabStemNew    = 0.
      RNStrucStemNew   = 0.
      RNStabTuberNew   = 0.
      RNStrucTuberNew  = 0.
      RNStabLeafLive    = 0.
      RNStrucLeafLive   = 0.
      RNStabStemLive    = 0.
      RNStrucStemLive   = 0.
      RNStabLeafDead    = 0.
      RNStrucLeafDead   = 0.
      RNStabStemDead    = 0.
      RNStrucStemDead   = 0.
      RNStabLeafLoss    = 0.
      RNStrucLeafLoss   = 0.
      RNStabStemLoss    = 0.
      RNStrucStemLoss   = 0.
      RNStabTuberLive   = 0.
      RNStrucTuberLive  = 0.
      RNStabTuberDead   = 0.
      RNStrucTuberDead  = 0.
      RNSol             = 0.
      RNSolDyingLeaves  = 0.
      leafclcntlast = 0
   else if (xNewTask == 'do_rates') then
      if (DateCmp2 < 1) then
         RNStabLeafLive    = 0.
         RNStrucLeafLive   = 0.
         RNStabStemLive    = 0.
         RNStrucStemLive   = 0.
         RNStabTuberLive   = 0.
         RNStrucTuberLive  = 0.
         RNSol             = 0.
         RNSolDead         = 0.
         RNSolLoss         = 0.
         RNSolDyingLeaves  = 0.
```

```
      RNStabLeafDead    = 0.
      RNStrucLeafDead   = 0.
      RNStabStemDead    = 0.
      RNStrucStemDead   = 0.
      RNStabLeafLoss    = 0.
      RNStrucLeafLoss   = 0.
      RNStabStemLoss    = 0.
      RNStrucStemLoss   = 0.
      if (LeafClCnt > 0) then
         call getsrtm (xMod, 'LeafArGrowth'    , LeafArGrowth, 0.)
         call getsrtm (xMod, 'LeafWtGrowth'    , LeafWtGrowth, 0.)
         call getsrtm (xMod, 'TuberWtGrowth'   , TuberWtGrowth, 0.)
         call getsrtm (xMod, 'StemWtGrowth'    , StemWtGrowth, 0.)
         call getsrtm (xMod, 'StemVolGrowth'   , StemVolGrowth, 0.)
         TotWtGrowth = LeafWtGrowth + StemWtGrowth + TuberWtGrowth
         call getsrt (xMod, 'StressIndexStAg'  , StressIndexStAg)
         call getsrt (xMod, 'StressIndexRoAg'  , StressIndexRoAg)
         call getsrt (xMod, 'StressIndexTuAg'  , StressIndexTuAg)
         call getsrt (xMod, 'StressIndexStAgP' , StressIndexStAgP)
         call getsrt (xMod, 'StressIndexRoAgP' , StressIndexRoAgP)
         call getsrt (xMod, 'StressIndexTuAgP' , StressIndexTuAgP)
         call getsrtm (xMod, 'NuptR', NuptR, 0.)
         call getsrt (xMod, 'FrReserveUse'    , FrReserveUse)
         call getsrt (xMod, 'FrRedistrUse'    , FrRedistrUse)
         call getsrt (xMod, 'ReserveLeafLive' , ReserveLeafLive)
         call getsrt (xMod, 'ReserveStemLive' , ReserveStemLive)
         call getsrt (xMod, 'ReserveTuberLive', ReserveTuberLive)
         call getsrt (xMod, 'StrucDMTuberLive'  , StrucDMTuberLive)
         RNStabLeafNewMax  = (LeafArGrowth/10000.) * NstabLeafMax
         RNStabStemNewMax  = StemVolGrowth  * NStabStemMax
         RNStabTuberNewMax = TuberWtGrowth  * NStabTuberMax
         RNStabLeafNewMin  = (LeafArGrowth/10000.) * NstabLeafMin
         RNStabStemNewMin  = StemVolGrowth  * NStabStemMin
         RNStabTuberNewMin = TuberWtGrowth  * NStabTuberMin
         RNStrucLeafNew  = (LeafArGrowth/10000.) * NstrucLeafMin
         RNStrucStemNew    = StemVolGrowth  * NStrucStemMin
         RNStrucTuberNew   = TuberWtGrowth  * NStrucTuberMin
         RNTotStabMax   = RNStabLeafNewMax + RNStabStemNewMax + &
            RNStabTuberNewMax
         RNTotStabMin   = RNStabLeafNewMin + RNStabStemNewMin + &
            RNStabTuberNewMin
         RNTotStruc     = RNStrucLeafNew + RNSTrucStemNew + &
            RNStrucTuberNew
         RNTotGrowthMax = RNTotStabMax + RNTotStruc
         RNTotGrowthMin = RNTotStabMin + RNTotStruc
         RNTuberResp = NStabTuberLive * max(0., StrucDMTuberLive  * &
            RelRespTuber ) / max(1.e-10, TuberWt)
         RedistrLeafNDead = 0.
         ReserveLeafNLive = 0.
         RNSolDyingLeaves = 0.
         do i = 1, leafclcnt
            if ((not(LeafAliveCl(i))).and.(NStabLiveCl(i)>1.e-8)) then
               RedistrLeafNDead    = RedistrLeafNDead + NStabLiveCl(i)
            elseif (LeafAliveCl(i)) then
               RNstabDeadCl(i)=NStabLiveCl(i) * StressIndexLvDyingCl(i)
               RNstrucDeadCl(i)=NStabLiveCl(i)* StressIndexLvDyingCl(i)
               ReserveLeafNLive=ReserveLeafNLive + max(0., &
                  (NStabLiveCl(i) -RNStabDeadCl(i) - NStabLeafMin * &
                  LeafArLiveCl(i)/10000.)) * kdestabLeaf
               RNSolDyingLeaves    = RNSolDyingLeaves + NSol * &
                  StressIndexLvDyingCl(i) * NStabLiveCl(i) / &
                  max(1.e-10, NStabLeafLive)
            end if
         end do
         ReserveStemNLive = (1.-(StressIndexStAg + StressIndexStAgP)) &
           * max(0., (NStabStemLive-NStabStemMin*StemVol))*kdestabStem
         ReserveTuberNLive = (1.-(StressIndexTuAg +StressIndexTuAgP)) &
           * max(0., (NStabTuberLive - NStabTuberMin * TuberWt - &
```

```
338              RNTuberResp)) * kdestabTuber
339
340            RedistrStemNDead = (StressIndexStAg + StressIndexStAgP) * &
341               NStabStemLive
342            RedistrTuberNDead = (StressIndexTuAg + StressIndexTuAgP) * &
343               NStabTuberLive
344            RedistrNTot  = RedistrLeafNDead + RedistrStemNDead + &
345               RedistrTuberNDead
346            ReserveNTot  = ReserveLeafNLive + ReserveStemNLive + &
347               ReserveTuberNLive
348            NAvailGrowth = ksynstab * NSol - RNTotGrowthMin + RNTuberResp
349            FrNReserveUse = 0.
350            FrNRedistrUse = 0.
351            redisN = RedistrNTot
352            reserN = ReserveNTot
353            if (NAvailGrowth < 0.) then
354               if((RedisN + ReserN + NavailGrowth)/max(1.e-10, &
355                  RNTotGrowthMin) < -1.e-6) then
356                  write(*, 2222)
357    2222          format ('Not enough N available for structural and &
358                           minimal stable N; check NDEMnPOS')
359               end if
360               FrNRedistrUse = min(1., max(0., -NAvailGrowth / &
361                  max(1.e-10,RedisN)))
362               NAvailGrowth = NAvailGrowth + FrNRedistrUse * RedisN
363               RedisN = max(0., (1. - FrNRedistrUse) * RedisN)
364               FrNReserveUse = min(1., max(0., -NAvailGrowth/max(1.e-10, &
365                  ReserN)))
366               NAvailGrowth = NAvailGrowth + FrNReserveUse * ReserN
367               if ((ReserN > 0.) .and. (NAvailGrowth < 0.)) then
368                  ReserN=max(0.,(1.-FrNReserveUse)*ReserN)+NAvailGrowth
369                  NAvailGrowth = 0.
370               else
371                  ReserN = max(0., (1. - FrNReserveUse) * ReserN)
372               end if
373            end if
374            if (NAvailGrowth < RNTotGrowthMax - RNTotGrowthMin) then
375               NbalGr = NAvailGrowth + redisN + reserN
376               if (NBalGr < 1.e-8) then
377                  RNStabLeafNew   = RNStabLeafNewMin
378                  RNStabStemNew   = RNStabStemNewMin
379                  RNStabTuberNew  = RNStabTuberNewMin
380                  RedisN = 0.
381                  ReserN = 0.
382               elseif ((RNTotGrowthMax - RNTotGrowthMin > 1.e-8) .and. &
383                  (NBalGr < RNTotGrowthMax - RNTotGrowthMin)) then
384                  FrNAvail = max(0., min(1., NBalGr / (RNTotGrowthMax - &
385                     RNTotGrowthMin)))
386                  RNStabLeafNew   = RNStabLeafNewMin  + FrNavail * &
387                     (RNStabLeafNewMax - RNStabLeafNewMin)
388                  RNStabStemNew   = RNStabStemNewMin  + FrNavail * &
389                     (RNStabStemNewMax - RNStabStemNewMin)
390                  RNStabTuberNew  = RNStabTuberNewMin + FrNavail * &
391                     (RNStabTuberNewMax - RNStabTuberNewMin)
392                  RedisN = 0.
393                  ReserN = 0.
394               else
395                  FrNRedistrUse = min(1., max(0., (RNTotGrowthMax - &
396                      RNTotGrowthMin - NAvailGrowth ) / max(1.e-8, &
397                      RedisN)))
398                  if (NAvailGrowth + FrNRedistrUse * RedisN < &
399                      RNTotGrowthMax - RNTotGrowthMin) then
400                     FrNReserveUse = min(1., max(0., (RNTotGrowthMax - &
401                        RNTotGrowthMin - NAvailGrowth - FrNRedistrUse * &
402                        RedisN) / max(1.e-8, ReserN)))
403                  end if
404                  RNStabLeafNew   = RNStabLeafNewMax
405                  RNStabStemNew   = RNStabStemNewMax
```

```
406                RNStabTuberNew  = RNStabTuberNewMax
407                RedisN = max(0., min(1., (1. - FrNRedistrUse))) * RedisN
408                ReserN = max(0., min(1., (1. - FrNReserveUse))) * ReserN
409             end if
410          else
411             RNStabLeafNew   = RNStabLeafNewMax
412             RNStabStemNew   = RNStabStemNewMax
413             RNStabTuberNew  = RNStabTuberNewMax
414          end if
415          FrNRedistrUse = max(0., min(1., (RedistrNTot - RedisN) / &
416             max(1.e-10, RedistrNTot)))
417          FrNReserveUse = max(0., min(1., (ReserveNTot - ReserN) / &
418             max(1.e-10, ReserveNTot)))
419          if (LeafArGrowth > 1.e-8) then
420             RNStrucLiveCl(LeafClCnt + 1) = RNStrucLeafNew
421             RNStabLiveCl(LeafClCnt + 1)  = RNStabLeafNew
422             RNStrucDeadCl(LeafClCnt + 1) = 0.
423             RNStabDeadCl(LeafClCnt + 1) = 0.
424             RNStrucLossCl(LeafClCnt + 1) = 0.
425             RNStabLossCl(LeafClCnt + 1) = 0.
426          else
427             RNStrucLiveCl(LeafClCnt + 1) = 0.
428             RNStabLiveCl(LeafClCnt + 1) = 0.
429             RNStrucDeadCl(LeafClCnt + 1) = 0.
430             RNStabDeadCl(LeafClCnt + 1) = 0.
431             RNStrucLossCl(LeafClCnt + 1) = 0.
432             RNStabLossCl(LeafClCnt + 1) = 0.
433          end if
434          RNSol = NuptR - RNStabLeafNew - RNStrucLeafNew -RNStabStemNew&
435              - RNStrucStemNew  - RNStabTuberNew - RNStrucTuberNew + &
436              FrNReserveUse * ReserveNTot + FrNRedistrUse * RedistrNTot&
437              + RNTuberResp - RNSolDyingLeaves
438          if ((NSol + RNSol) / max(1.e-8,abs(NSol)) < -1.e-6 ) then
439             if((abs(NSol) <= 1.e-6) .and. (abs(RNSol) <= 1.e-6)) then
440                RNSol = - NSol
441             else
442                write(*,*) NSol, RNSol
443             endif
444          endif
445          RNStabLeafLive    = RNStabLeafNew
446          RNStrucLeafLive   = RNStrucLeafNew
447          do i = 1, LeafClCnt
448             RNStabLiveCl(i)  = 0.
449             RNStabDeadCl(i)  = 0.
450             RNStabLossCl(i)  = 0.
451             RNStrucDeadCl(i) = 0.
452             RNStrucLiveCl(i) = 0.
453             RNStrucLossCl(i) = 0.
454             if ((not(LeafAliveCl(i))).and.(NStabLiveCl(i)> 1.e-8)) then
455                RNStabLiveCl(i)  = - NStabLiveCl(i)
456                RNStrucLiveCl(i) = - NStrucLiveCl(i)
457                RNStabDeadCl(i)  =   NStabLiveCl(i) - FrNRedistrUse * &
458                   NstabLiveCl(i)
459                RNStrucDeadCl(i) =   NStrucLiveCl(i)
460             elseif(not(LeafAliveCl(i))) then
461                if (NStabDeadCl(i) > 1.e-8) then
462                   RNStabDeadCl(i) = -NStabDeadCl(i) * RelLossDeadLeaves
463                   RNStabLossCl(i) =  NStabDeadCl(i) * RelLossDeadLeaves
464                else
465                   NStabDeadCl(i) = 0.
466                end if
467                if (NStrucDeadCl(i) > 1.e-8) then
468                   RNStrucDeadCl(i) =-NStrucDeadCl(i)* RelLossDeadLeaves
469                   RNStrucLossCl(i) = NStrucDeadCl(i)* RelLossDeadLeaves
470                else
471                   NStrucDeadCl(i) = 0.
472                end if
473             else
```

```
474                RNstabDeadCl(i)= NStabLiveCl(i)* StressIndexLvDyingCl(i)
475                RNStabLiveCl(i)= -FrNReserveUse * max(0.,NStabLiveCl(i)&
476                  - RNStabDeadCl(i) - NStabLeafMin * LeafArLiveCl(i) / &
477                    10000.) * kdestabLeaf
478                RNstrucDeadCl(i)=NStrucLiveCl(i)*StressIndexLvDyingCl(i)
479                RNstrucLiveCl(i)= -RNStrucDeadCl(i)
480             end if
481             RNStabLeafLive  = RNStabLeafLive  + RNStabLiveCl(i)
482             RNStabLeafDead  = RNStabLeafDead  + RNStabDeadCl(i)
483             RNStabLeafLoss  = RNStabLeafLoss  + RNStabLossCl(i)
484             RNStrucLeafLive = RNStrucLeafLive + RNStrucLiveCl(i)
485             RNStrucLeafDead = RNStrucLeafDead + RNStrucDeadCl(i)
486             RNStrucLeafLoss = RNStrucLeafLoss + RNStrucLossCl(i)
487          end do
488          if (RNStabLeafDead >= NStabLeafLive) then
489             RNSolDead = NSol
490          end if
491          if (RNStabLeafLoss >= NStabLeafDead) then
492             RNSolLoss = NSolDead
493          end if
494          RNStabStemLive = RNStabStemNew  - FrNReserveUse * &
495             ReserveStemNLive  - max(0., min(1., (StressIndexStAg + &
496             StressIndexStAgP))) * NStabStemLive
497          RNStabTuberLive = RNStabTuberNew  - FrNReserveUse * &
498             ReserveTuberNLive - max(0., min(1., (StressIndexTuAg + &
499             StressIndexTuAgP))) * NStabTuberLive - RNTuberResp
500          RNStrucStemLive = RNStrucStemNew  - max(0., min(1., &
501             (StressIndexStAg + StressIndexStAgP))) * NStrucStemLive
502          RNStrucTuberLive  = RNStrucTuberNew - max(0., min(1., &
503             (StressIndexTuAg + StressIndexTuAgP))) * NStrucTuberLive
504          RNStabStemDead = max(0., min(1., (StressIndexStAg + &
505              StressIndexStAgP))) * NStabStemLive  - FrNRedistrUse * &
506              RedistrStemNDead  - NStabStemDead  * RelLossDeadStems
507          RNStabTuberDead = max(0., min(1., (StressIndexTuAg + &
508             StressIndexTuAgP))) * NStabTuberLive - FrNRedistrUse * &
509             RedistrTuberNDead - NStabTuberDead * RelLossDeadTubers
510          RNStrucStemDead   = max(0., min(1., (StressIndexStAg + &
511             StressIndexStAgP))) * NStrucStemLive  - NStrucStemDead * &
512             RelLossDeadStems
513          RNStrucTuberDead  = max(0., min(1., (StressIndexTuAg + &
514             StressIndexTuAgP))) * NStrucTuberLive - NStrucTuberDead * &
515             RelLossDeadTubers
516          RNStabStemLoss    = NStabStemDead   * RelLossDeadStems
517          RNStabTuberLoss   = NStabTuberDead  * RelLossDeadTubers
518          RNStrucStemLoss   = NStrucStemDead  * RelLossDeadStems
519          RNStrucTuberLoss  = NStrucTuberDead * RelLossDeadTubers
520       endif
521    endif
522 else if (xNewTask == 'output') then
523    call outdat (2,0,'ProtTCalc'        , ProtTCalc)
524    call outdat (2,0,'ProtRCalc'        , ProtRCalc)
525 else if (xNewTask == 'do_states') then
526    NstabconcLeaf   = 0.
527    NstabconcLeafWt = 0.
528    Norgconcleafwt  = 0.
529    Nitrconcleafwt  = 0.
530    NstrucconcLeaf  = 0.
531    NstrucconcLeafWt= 0.
532    NtotconcLeaf    = 0.
533    NtotconcLeafWt  = 0.
534    NstabconcStem   = 0.
535    NstabconcStemVol= 0.
536    NOrgConcStem    = 0.
537    NitrConcStem    = 0.
538    NstrucconcStem  = 0.
539    NtotconcStem    = 0.
540    NstabconcTuber  = 0.
541    NstrucconcTuber = 0.
```

```
      NOrgConcTuber   = 0.
      NitrConcTuber   = 0.
      NtotconcTuber   = 0.
      NtotconcTotal   = 0.
      NtotconcAboveWt = 0.
      NOrgConcAboveWt = 0.
      NitrConcAboveWt = 0.
      NtotconcAboveAr = 0.
      NtotconcUnder   = 0.
      nsolconc        = 0.
      NStabConcLeafLive = 0.
      NStabConcLeafDead = 0.
      NStabConcLeafLoss = 0.
      ProtRCalc    = 0.
      ProtTCalc    = 0.
      NStabAdd = 0.
      FreeDmLeaf = 0.
      NStabMinLeaf = 0.
      nstabconcleafphot = 0.
      ntotconcleafwtphot = 0.
      SPADCalc = 0.
      LaiSpad = 0.
      NSpadConcLeafWt = 0.
      FreeDMStemLive = 0.
      FreeDMTuberLive = 0.
      NSolConc      = 0.
      NTotConcTotal = 0.
      NTotConcAboveWt = 0.
      NTotConcAboveAr = 0.
      NTotConcUnder = 0.
      SPADCalc = 0.
      if (leafclcnt > 0) then
         do i = 1, LeafClCnt
            NSolLiveCl(i)   = 0.
            NStabClassConc(i) = 0.
            NStrucClassConc(i) = 0.
         end do
      end if
      call getsi (xMod, 'DateCmp', DateCmp)
      call getsi (xMod, 'DateCmp2', DateCmp2)
      call getsi (xMod, 'DateCmp3', DateCmp3)
      if (DateCmp2 < 1) then
         call getsrt (xMod, 'leafwt' , leafwt)
         call getsrt (xMod, 'leafdeadwt' , leafdeadwt)
         call getsrt (xMod, 'lailive' , lailive)
         call getsrt (xMod, 'laidead' , laidead)
         call getsrt (xMod, 'lailoss' , lailoss)
         call getsrt (xMod, 'stemwt' , stemwt)
         call getsrt (xMod, 'stemDeadwt' , stemDeadwt)
         call getsrt (xMod, 'tuberwt', tuberwt)
         call getsrt (xMod, 'TuberDeadwt' , TuberDeadwt)
         totaldrywtpresent = leafwt + LeafDeadWt + stemwt + StemDeadWt + &
             tuberwt + TuberDeadWt
         totaldrywt = leafwt + stemwt + tuberwt
         call getsrt (xMod, 'stemvol' , stemvol)
         call getsi (xMod, 'leafclcnt', leafclcnt)
         if (DateCmp ==  0) then
            RNStabLeafLive   = LaiLive  * NstabLeafMax
            RNStrucLeafLive  = LaiLive  * NstrucLeafMin
            if (LeafClCnt > 0) then
               do i = 1, LeafClCnt
                  RNStrucLiveCl(i) = RNStrucLeafLive / (1.*LeafClCnt)
                  RNStabLiveCl(i)  = RNStabLeafLive  / (1.*LeafClCnt)
               end do
            end if
            RNStabStemLive   = StemVol * NstabStemMax
            RNStrucStemLive  = StemVol * NStrucStemMin
            RNStabTuberLive  = TuberWt * NstabTuberMax
```

```
610            RNStrucTuberLive = TuberWt * NStrucTuberMin
611            RNSol            = NSolConcI * TotalDryWtPresent
612            NinLeafI = RNStabLeafLive + RNStrucLeafLive + RNSol * &
613               LeafWt/(LeafWt + StemWt + RootWt)
614            NinStemI = RNStabStemLive + RNStrucStemLive + RNSol * &
615               StemWt/(LeafWt + StemWt + RootWt)
616            NinTuberI = RNStabTuberLive + RNStrucTuberLive
617         end if
618         if(DateCmp >= 0) then
619            do i = LeafClCnt, 1, -1
620               NStabDeadCl(i)=intgrl(NStabDeadCl(i),RNStabDeadCl(i),delt)
621               NStabLossCl(i)=intgrl(NStabLossCl(i),RNStabLossCl(i),delt)
622               NStabLiveCl(i)=intgrl(NStabLiveCl(i),RNStabLiveCl(i),delt)
623               NStrucLiveCl(i) = intgrl(NStrucLiveCl(i),RNStrucLiveCl(i),&
624                  delt)
625               NStrucDeadCl(i) = intgrl(NStrucDeadCl(i),RNStrucDeadCl(i),&
626                  delt)
627               NStrucLossCl(i) = intgrl(NStrucLossCl(i),RNStrucLossCl(i),&
628                  delt)
629            end do
630            LeafClCntLast = LeafClCnt
631            NstabLeafLive = intgrl (NstabLeafLive, RNstabLeafLive , delt)
632            NstabStemLive = intgrl (NstabStemLive, RNstabStemLive , delt)
633            NstabTuberLive = intgrl(NstabTuberLive, RNstabTuberLive, delt)
634            NstrucLeafLive  = intgrl(NstrucLeafLive,RNstrucLeafLive, delt)
635            NstrucStemLive  = intgrl(NstrucStemLive,RNstrucStemLive, delt)
636            NstrucTuberLive =intgrl(NstrucTuberLive,RNstrucTuberLive,delt)
637            NstabLeafDead   =intgrl(NstabLeafDead, RNstabLeafDead, delt)
638            NstabStemDead   = intgrl(NstabStemDead, RNstabStemDead, delt)
639            NstabTuberDead  = intgrl(NstabTuberDead,RNstabTuberDead,delt)
640            NstabLeafLoss   = intgrl(NstabLeafLoss, RNStabLeafLoss, delt)
641            NstabStemLoss   = intgrl(NstabStemLoss, RNStabStemLoss, delt)
642            NstabTuberLoss = intgrl(NstabTuberLoss, RNStabTuberLoss, delt)
643            NstrucLeafDead = intgrl(NstrucLeafDead, RNstrucLeafDead, delt)
644            NstrucStemDead = intgrl(NstrucStemDead, RNstrucStemDead, delt)
645            NstrucTuberDead= intgrl(NstrucTuberDead,RNstrucTuberDead,delt)
646            NstrucLeafLoss = intgrl(NstrucLeafLoss,RNstrucLeafLoss, delt)
647            NstrucStemLoss = intgrl(NstrucStemLoss,RNstrucStemLoss, delt)
648            NstrucTuberLoss= intgrl(NstrucTuberLoss,RNstrucTuberLoss,delt)
649            NSol           = intgrl(NSol, RNSol - RNSolDead, delt)
650            NSolDead       = intgrl(NSolDead, RNSolDead - RNSolLoss, delt)
651            NSolLoss       = intgrl(NsolLoss, RNSolLoss, delt)
652            NUptakeTot     = intgrl(NUptakeTot, NUptR, delt)
653         end if
654         NtotLeafLive = NstabLeafLive + NstrucLeafLive + NSol * LeafWt / &
655            max(1.e-8,(LeafWt + StemWt ))
656         NtotLeafDead  = NstabLeafDead  + NstrucLeafDead
657         NtotLeafLoss  = NstabLeafLoss  + NstrucLeafLoss
658         NtotStemLive = NstabStemLive + NstrucStemLive + NSol * StemWt / &
659            max(1.e-8,(LeafWt + StemWt ))
660         NtotStemDead  = NstabStemDead  + NstrucStemDead
661         NtotStemLoss  = NstabStemLoss  + NstrucStemLoss
662         NtotTuberLive = NstabTuberLive + NstrucTuberLive
663         NtotTuberDead = NstabTuberDead + NstrucTuberDead
664         NtotTuberLoss = NstabTuberLoss + NstrucTuberLoss
665         NtotAboveLive = NtotLeafLive   + NtotSTemLive
666         NtotAboveDead = NtotLeafDead   + NtotSTemDead
667         NtotAboveLoss = NtotLeafLoss   + NtotSTemLoss
668         NtotUnderLive = NtotTuberLive
669         NtotUnderDead = NtotTuberDead
670         NtotUnderLoss = NtotTuberLoss
671         NtotTotalLive = NTotAboveLive  + NtotUnderLive
672         NtotTotalDead = NTotAboveDead  + NtotUnderDead + NSolDead
673         NtotTotalLoss = NTotAboveLoss  + NtotUnderLoss + NSolLoss
674         NNorgLeaf = NStabLeafLive + NStrucLeafLive + NStabLeafDead + &
675            NStrucLeafDead
676         NNorgStem = NStabStemLive + NStrucStemLive + NStabStemDead + &
677            NStrucStemDead
```

```
      NNorgTuber = NStabTuberLive + NStrucTuberLive + NStabTuberDead +&
         NStrucTuberDead
      NNorgAbovewt   = NNorgLeaf    + NNorgStem
      NNorgTotal     = NNorgAboveWt + NNorgTuber
      NNitrLeaf = NSol * LeafWt/max(1.e-8,(LeafWt + StemWt ))
      NNitrStem = NSol * StemWt/max(1.e-8,(LeafWt + StemWt ))
      NNitrTuber = 0.
      NNitrAbovewt   = NNitrLeaf    + NNitrStem
      NNitrTotal     = NNitrAboveWt + NNitrTuber
      if (NtotTuberLive + NtotTuberDead > 1.e-8) then
         ProtRCalc = alphaProtR * &
            ((NtotTuberLive + NtotTuberDead)**betaProtR)
         ProtTCalc = alphaProtT * &
            ((NtotTuberLive + NtotTuberDead)**betaProtT)
      end if
      NtotIn = NUptakeTot + NinLeafI + NinStemI + NinTuberI
      NtotAccounted = NtotTotalLive + NtotTotalDead + NtotTotalLoss
      Nbalance = NtotIn - NtotAccounted
      if (abs(Nbalance)/max(1.e-8, abs(NtotIn + NtotAccounted)) > &
         1.e-3) then
          write(*,*) Nbalance, NtotIn, NtotAccounted
      end if
      sumtrans= 0.
      LeafAr = 0.
      LeafDeadAr = 0.
      if (leafClCnt > 0) then
         do i = LeafClCnt, 1, -1
            if (i < LeafClCnt) LeafAr = LeafAr + LeafArLiveCl(i+1) / &
               10000.
            if (i < LeafClCnt) LeafDeadAr = LeafDeadAr + &
               LeafArDeadCl(i+1) / 10000.
            Partransfraction(i) = LeafArLiveCl(i) * exp (max(-30. , &
               min(30.,-ECPDF * (LeafAr + relLIntleafdead *LeafDeadAr)&
               * NDilutionPar)))
            sumtrans = sumtrans + partransfraction(i)
         end do
         do i = 1, LeafClCnt
            Partransfraction(i) = max(0., min(1.,partransfraction(i) /&
               max(1.e-6,sumtrans)))
            Partrans(i) = Partransfraction(i)
            if (LeafAliveCl(i)) then
               NStabConcLeafPhot  = NStabConcLeafPhot + Partrans(i) * &
                  NStabLiveCl(i)/max(1.e-8,LeafArLiveCl(i)/10000.)
               NTotConcLeafWtPhot = NTotConcLeafWtPhot + Partrans(i) *&
                  (NStabLiveCl(i)+NStrucLiveCl(i)) / max(1.e-8, &
                  LeafWtLiveCl(i))
               NSolLiveCl(i) = NSol * LeafWtLiveCl(i) / max(1.e-8, &
                  (LeafWt + StemWt ))
               NStabClassconc(i) = NStabLiveCl(i) / max(1.e-8, &
                  LeafWtLiveCl(i))
               NStrucClassconc(i) = NStrucLiveCl(i) / max(1.e-8, &
                  LeafWtLiveCl(i))
            end if
         end do
      end if
      if (LeafClCnt > 0) then
         do i = 1, leafclcnt
            FreeDMLeafCl(i) = 0.
         end do
      end if
      if (LeafWt + LeafDeadWt > 1.e-8) then
         NstabconcLeafWt = (NstabLeafLive + NStabLeafDead)  / &
            (LeafWt + LeafDeadWt)
         NstrucconcLeafWt= (NstrucLeafLive + NStrucLeafDead) / &
            (LeafWt + LeafDeadWt)
         NOrgConcLeafWt  = (NStabLeafLive + NStabLeafDead + &
            NStrucLeafLive + NstrucLeafDead) /  (LeafWt + LeafDeadWt)
         NtotconcLeafWt  = (NtotLeafLive + NtotLeafDead) / (&
```

```
746               LeafWt + LeafDeadWt)
747             NitrConcLeafWt  = NtotConcLeafWt - NOrgConcLeafWt
748             if (lailive > 1.e-8) then
749                NStabConcLeafLive = NStabLeafLive / max(1.e-8, LaiLive)
750             endif
751             if (laidead > 1.e-8) then
752                NStabConcLeafDead = NStabLeafDead / max(1.e-8, LaiDead)
753             endif
754             if (lailoss > 1.e-8) then
755                NStabConcLeafLoss = NStabLeafLoss / max(1.e-8, LaiLoss)
756             endif
757             if (lailive+laidead > 1.e-8) then
758                NstrucconcLeaf  = (NstrucLeafLive + NStrucLeafDead) / &
759                   (LaiLive + LaiDead)
760                NtotconcLeaf    = (NtotLeafLive + NtotLeafDead) / &
761                   (LaiLive + LaiDead)
762                NstabconcLeaf   = (NstabLeafLive + NStabLeafDead) / &
763                   (LaiLive + LaiDead)
764             endif
765             if(NstabLeafLive > 1.e-8) then
766                do i = 1, leafclcnt
767                   if(leafalivecl(i)) then
768                      FreeDMLeafCl(i) = LeafWtLiveCl(i) - max(0., min( &
769                         LeafWtLiveCl(i), NStrucLiveCl(i)*DMNStrucLeaf + &
770                         NStabLeafMin*(LeafArLiveCl(i)/10000.)* &
771                         DMNStabLeafMin))
772                   end if
773                end do
774             endif
775          endif
776          if (StemWt + StemDeadWt > 1.e-8) then
777             NstabconcStem    = (NstabStemLive  + NStabStemDead)  / &
778               (StemWt + StemDeadWt)
779             NstrucconcStem   = (NstrucStemLive + NStrucStemDead) / &
780               (StemWt + StemDeadWt)
781             NOrgConcStem     = (NstabStemLive  + NStabStemDead + &
782                NstrucStemLive + NStrucStemDead)  / (StemWt + StemDeadWt)
783             NtotconcStem     = (NtotStemLive   + NtotstemDead)   / &
784               (StemWt + StemDeadWt)
785             NitrConcStem     = NTotConcStem - NOrgConcStem
786             if (StemVol > 1.e-8) then
787                NStabConcStemVol =  NStabStemLive / max(1.e-8, StemVol)
788             end if
789             if(NstabStemLive > 1.e-8) then
790                FreeDMStemLive = (StemWt  - max(0., min (StemWt, &
791                DMNStrucStem * NStrucStemLive  + NStabStemMin * StemVol * &
792                DMNStabStemMin)))
793             endif
794          endif
795          if (TuberWt + TuberDeadWt > 1.e-8) then
796             NstabconcTuber   = (NstabTuberLive  + NStabTuberDead)  / &
797                (TuberWt + TuberDeadWt)
798             NstrucconcTuber  = (NstrucTuberLive + NStrucTuberDead) / &
799                (TuberWt + TuberDeadWt)
800             NOrgConcTuber    = (NstabTuberLive  + NStabTuberDead + &
801                 NstrucTuberLive + NStrucTuberDead)  / (TuberWt + &
802                TuberDeadWt)
803             NtotconcTuber    = (NtotTuberLive   + NTotTuberDead)   / &
804                (TuberWt + TuberDeadWt)
805             NitrConcTuber    = NtotConcTuber - NOrgConcTuber
806             if(NstabTuberLive > 1.e-8 ) then
807                FreeDMTuberLive    = (TuberWt - max(0., min(TuberWt, &
808                   DMNStrucTuber * NStrucTuberLive + NStabTuberMin * &
809                   TuberWt*DMNStabTuberMin)))
810             endif
811          endif
812          if ((LeafWt + StemWt ) > 1.e-8) then
813             NSolconc = NSol / (LeafWt + StemWt )
```

```
814            NtotconcTotal =(NtotTotalLive+NtotTotalDead)/TotalDryWtPresent
815         end if
816         if (StemWt+LeafWt+StemDeadWt+LeafDeadWt > 1.e-8) then
817            NtotconcAboveWt = (NtotAboveLive + NtotAboveDead) /max(1.e-8,&
818                StemWt + LeafWt + StemDeadWt + LeafDeadWt)
819            NOrgConcAboveWt = (NstabLeafLive  + NStabLeafDead + &
820               NstabStemLive  + NStabStemDead + NStrucLeafLive  + &
821               NStrucLeafDead + NStrucStemLive  + NStrucStemDead) / &
822               max(1.e-8, StemWt + LeafWt + StemDeadWt + LeafDeadWt)
823            NtotconcAboveAr = (NtotAboveLive + NtotAboveDead) /max(1.e-8,&
824               LaiLive+LaiDead)
825            NitrConcAboveWt = NtotConcAboveWt - NOrgConcAboveWt
826            SPADcalc= kspad * NtotConcLeaf + bspad * max(0., 1.- exp(&
827               max(-30., min(30., -cspad * NtotConcLeaf))))
828         endif
829         if (TuberWt + TuberDeadWt > 1.e-8) then
830            NtotconcUnder   = (NtotUnderLive + NtotUnderDead) / &
831               max(1.e-8,TuberWt + TuberDeadWt)
832         endif
833      elseif (DateCmp2 == 0) then
834         NstabLeafLoss   = intgrl (NstabLeafLoss,   NStabLeafLive + &
835            NStabLeafDead,   delt)
836         NstabStemLoss   = intgrl (NstabStemLoss,   NStabStemLive + &
837            NStabStemDead,   delt)
838         NstrucLeafLoss  = intgrl (NstrucLeafLoss,  NstrucLeafLive + &
839            NStrucLeafDead,  delt)
840         NstrucStemLoss  = intgrl (NstrucStemLoss,  NstrucStemLive + &
841            NStrucStemDead,  delt)
842         NstabLeafLive   = intgrl (NstabLeafLive,   -NstabLeafLive , delt)
843         NstabStemLive   = intgrl (NstabStemLive,   -NstabStemLive , delt)
844         NstrucLeafLive = intgrl(NstrucLeafLive, -NstrucLeafLive, delt)
845         NstrucStemLive = intgrl(NstrucStemLive, -NstrucStemLive, delt)
846         NstabLeafDead  = intgrl(NstabLeafDead, -NstabLeafDead, delt)
847         NstabStemDead  = intgrl(NstabStemDead, -NstabStemDead, delt)
848         NstrucLeafDead = intgrl(NstrucLeafDead, -NstrucLeafDead, delt)
849         NstrucStemDead = intgrl(NstrucStemDead, -NstrucStemDead,  delt)
850         NSol     = intgrl (NSol,            -NSol     , delt)
851         NSolDead = intgrl (NSolDead,        -NSolDead , delt)
852         NSolLoss = intgrl (NsolLoss,        NSol + NSolDead , delt)
853         NtotLeafLive  = 0.
854         NtotLeafDead  = 0.
855         NtotLeafLoss  = 0.
856         NtotStemLive  = 0.
857         NtotStemDead  = 0.
858         NtotStemLoss  = 0.
859         NtotAboveLive = 0.
860         NtotAboveDead = 0.
861         NtotAboveLoss = 0.
862         NtotUnderLive = 0.
863         NtotUnderDead = 0.
864         NtotUnderLoss = 0.
865         NtotTotalLive = 0.
866         NtotTotalDead = 0.
867         NtotTotalLoss = 0.
868      end if
869      call putsrt (xMod, 'NStabLeafLive'    , NStabLeafLive)
870      call putsrt (xMod, 'NStabStemLive'    , NStabStemLive)
871      call putsrt (xMod, 'NStabTuberLive'   , NStabTuberLive)
872      call putsrt (xMod, 'NStrucLeafLive'   , NStrucLeafLive)
873      call putsrt (xMod, 'NStrucStemLive'   , NStrucStemLive)
874      call putsrt (xMod, 'NStrucTuberLive'  , NStrucTuberLive)
875      call putsrt (xMod, 'NTotLeafLive'     , NTotLeafLive)
876      call putsrt (xMod, 'NTotStemLive'     , NTotStemLive)
877      call putsrt (xMod, 'NTotTuberLive'    , NTotTuberLive)
878      call putsrt (xMod, 'NTotTotalLive'    , NTotTotalLive)
879      call putsrt (xMod, 'NTotAboveLive'    , NTotAboveLive)
880      call putsrt (xMod, 'NTotUnderLive'    , NTotUnderLive)
881      call putsrt (xMod, 'NTotConcLeaf'     , NTotConcLeaf)
```

```
      call putsrt (xMod, 'NTotConcLeafWt'   , NTotConcLeafWt)
      call putsrt (xMod, 'NTotConcStem'     , NTotConcStem)
      call putsrt (xMod, 'NTotConcTuber'    , NTotConcTuber)
      call putsrt (xMod, 'NTotConcAboveAr'  , NTotConcAboveAr)
      call putsrt (xMod, 'NTotConcAboveWt'  , NTotConcAboveWt)
      call putsrt (xMod, 'NTotConcUnder'    , NTotConcUnder)
      call putsrt (xMod, 'NTotConcTotal'    , NTotConcTotal)
      call putsrt (xMod, 'nsol'             , nsol)
      call putsrt (xMod, 'nsolconc'         , nsolconc)
      call putsrt (xMod, 'nstabconcleaf'    , nstabconcleaf)
      call putsrt (xMod, 'nstabconcleafLive', nstabconcleafLive)
      call putsrt (xMod, 'nstabconcleafwt'  , nstabconcleafWt)
      call putsrt (xMod, 'nstabconcstem'    , nstabconcstem)
      call putsrt (xMod, 'nstabconcstemVol' , nstabconcstemVol)
      call putsrt (xMod, 'nstabconctuber'   , nstabconctuber)
      call putsrt (xMod, 'nstabconcleafphot', nstabconcleafphot)
      call putsrt (xMod, 'ntotconcleafwtphot', ntotconcleafwtphot)
      call putsrt (xMod, 'FreeDMStemLive'   , FreeDMStemLive)
      call putsrt (xMod, 'FreeDMTuberLive'  , FreeDMTuberLive)
      call putsrt (xMod, 'SPADCalc', SPADCalc)
      call putsrt (xMod, 'NTotAbove' , NTotAboveLive + NTotAboveDead)
      call putsrt (xMod, 'NTotLeaf', NTotLeafLive  + NTotLeafDead)
      call putsrt (xMod, 'NTotStem', NTotStemLive  + NTotStemDead)
      call putsrt (xMod, 'NTotTuber', NTotTuberLive + NTotTuberDead)
      call putsrt (xMod, 'NTotTotal'   , NTotTotalLive + NTotTotalDead )
      call putsrt (xMod, 'ProtTCalc'   , ProtTCalc)
      call putsrt (xMod, 'ProtRCalc'   , ProtRCalc)
      call putsrt (xMod, 'NNitrLeaf', NNitrLeaf)
      call putsrt (xMod, 'NNitrStem', NNitrStem)
      call putsrt (xMod, 'NNitrTuber', NNitrTuber)
      call putsrt (xMod, 'NNitrAbovewt', NNitrAboveWt)
      call putsrt (xMod, 'NNitrTotal', NNitrTotal)
      call putsrt (xMod, 'NNorgLeaf', NNorgLeaf)
      call putsrt (xMod, 'NNorgStem', NNorgStem)
      call putsrt (xMod, 'NNorgTuber', NNorgTuber)
      call putsrt (xMod, 'NNorgAbovewt', NNorgAboveWt)
      call putsrt (xMod, 'NNorgTotal', NNorgTotal)
   return
   end subroutine NDem4Rea
```

I.25. Subroutine NUPTB

```
subroutine Nuptb(xNewTask, xMod, xLogFileUnit)
implicit none
include '..\GetSoilData\SoilDataSet.inc'
character*(*), intent(in) :: xNewTask,xMod
character*13 :: OldTask, EmptyString
integer, intent(in) ::       xLogFileUnit
integer :: I, INLAY, INCLAY, ITRLAY, itemp
real, parameter :: TC1 = 1., MaxFracNup = 0.95, tiny = 1.e-10
real, dimension(NlMxn) :: TRLAY, PNUP1, PNUP2, NUPTL, NupMaxL, pnupdif, &
   pnuptran
real :: CumNuptake, Transp, NdemT, NAvai1, NAvai2, NUPTR, Frac, XNO3, &
   maxnuprate, maxnupratemax, NuptMaxP, NupMaxLT, navail, ndif, ntran, &
   nsolconc, nupred, nsolm, delt, rootdepth
SAVE
if (xNewTask == 'initialize') then
   call Logmessage (xLogFileUnit, 9, xMod, '1.0')
   Call getsrp (xMod, 'MaxNupRatemax', MaxNupRatemax)
   call getsrp (xMod, 'delt', delt)
   CumNuptake = 0.
   Nuptr = 0.
   do i = 1, Nl
      Nuptl(i) = 0.
   end do
else if (xNewTask == 'do_rates') then
   call getsrt (xMod,'TransAct',Transp)
   if (transp < 0.) transp = -transp
   if (Transp > tiny) then
      CALL GETART (xMod, 'Trwl'  , TRLAY , Nl , ITRLAY)
      call GETSRT (xMod, 'NSolConc', NSolConc)
      call GETSRT (xMod, 'NSol', NSol)
      call GETSRT (xMod, 'NuptMaxP', NuptMaxP)
      call getsrt (xMod, 'RootDepth',RootDepth)
      NDIF = 0.
      NTRAN = 0.
      NAVAIL = 0.
      NUpRed = 1.
      MAXNUPRATE = 1.
      NupMaxLT   = 0.
      DO I=istart,Nl
         Nuptl(i) = 0.
         if (TrLay(i) < 0.) TrLay(i) = -TrLay(i)
         nupred = max(0., min(1., 1. - exp(max(-30.,min(30., -MaxNupRateMax * &
            TrLay(i) / max(1.e-10, frvolumelayer(i) * TKL(i) * WCLQT(i) *1000.&
            ))))))
         PNUPDIF(I) = ANLAY(I) * nupred
         NDIF  = NDIF + PNUPDIF(I)
         PNUPTRAN(I) = TRLAY(I) * NCLAY(I) * 1.e7
         NTRAN  = NTRAN + PNUPTRAN(I)
         NAVAIL = NAVAIL+ANLAY(I)
         NupMaxL(i) = max(PnupDif(i), PnupTran(i), 0.)
         NupMaxLT   = NupMaxLT + NupMaxL(i)
      ENDDO
      NDEMT = max(min(NuptMaxP, NupMaxLT), 0.)
      NUPTR=0.
      DO I=istart, NL
         FRAC = NupMaxL(i) / max(1.e-8, NupMaxLT)
         NUPTL(I) = min(Frac*NDEMT,ANLAY(I))
         NUPTR   = NUPTR + NUPTL(I)
         NAVAIL  = NAVAIL- NUPTL(I)
      ENDDO

   else
      do i = 1, nl
         NUPTL(I) = 0.
      end do
```

```
66          NUPTR   = 0.
67          NDEMT = 0.
68       end if
69       CumNuptake = CumNuptake + Nuptr
70       CALL PUTSRT(xMod, 'NUPTR' , NUPTR )
71       CALL PUTART(xMod, 'NUPTL'  , NUPTL  , Nl)
72    else if (xNewTask == 'do_states') then
73       CumNuptake = CumNuptake + Nuptr
74       CALL PUTSRT (xMod,'NUPTOT' , CumNuptake)
75    end if
76    end subroutine Nuptb
```

I.26. Subroutine STOCKDAMAGE_DYNAMIC

```
subroutine stockdamage_dynamic (xNewTask, xMod,xLogFileUnit)
implicit none
include '..\GetCropData\CropDataSet.inc'
integer :: iset, CNamesCnt3, I, IFindC, InTb, ITb50, stocktype, DurStock,&
   DateCmp3, xLogFileUnit
parameter (ITb50=50)
character*(*) :: xNewTask,xMod
character(len=13) :: OldTask, EmptyString
character*80 CNames3(ITb50)
real :: Alpha_BI, Beta_BI, DamageIndex_Harvest, UBGRespLoss, DMRespLoss, &
   StarchRespLoss, StockDur, StockTemp, AlphaST, StockTempb, StockTempSum, &
   StockTempSumChange, AvRespLossPerc, DamageLossPerc, StockLossDryWt, &
   StockLossFreshWt, UBGStockLossPerc, DMStockLossPerc, StarchStockLossPerc, &
   UnavUBGRespLossPerc, UnavDMRespLossPerc, UnavStarchRespLossPerc, &
   RDamageLossPerc, RAvLoss, UBGStock, UBGUnavResp, UBGAvResp, StarchStock, &
   StarchUnavResp, StarchAvResp, TuberWtStock, TuberWtUnavResp, TuberWtAvResp,&
   UBGCalc, StarchCalc, X, Delt, Doy
real, dimension(ITb50) :: AlphaBI, BetaBI, UBGRespL, DMRespL, StarchRespL
save
if (xNewTask == 'initialize') then
   call Logmessage (xLogFileUnit,9,xMod,'0.0')
   call getac(xMod, 'CNames3', CNames3, CNamesMxn, CNamesCnt3)
   Do I=1,CNamesCnt3
      call lowerc(CNames3(i))
   EndDo
   call getarp (xMod,  'UBGRespL',    UBGRespL,    ITb50, CNamesCnt3)
   call getarp (xMod,  'DMRespL',     DMRespL,     ITb50, CNamesCnt3)
   call getarp (xMod,  'StarchRespL', StarchRespL, ITb50, CNamesCnt3)
   call getarp (xMod,  'AlphaBI' ,    AlphaBI ,    ITb50, CNamesCnt3)
   call getarp (xMod,  'BetaBI'  ,    BetaBI  ,    ITb50, CNamesCnt3)
   call getsrp (xMod, 'RAvLoss', RAvLoss)
   call getsc (xMod,'CName',CName)
   call lowerc(Cname)
   i = ifindc (CNames3,CNamesMxn,1,CNamesCnt3,CName)
   if (i<=0) i = CNamesCnt3
   UBGRespLoss    = UBGRespL(i)
   DMRespLoss     = DMRespL(i)
   StarchRespLoss = StarchRespL(i)
   if (UBGRespLoss < 0.)    UBGRespLoss    = UBGRespL(CNamesCnt3)
   if (DMRespLoss < 0.)     DMRespLoss     = DMRespL(CNamesCnt3)
   if (StarchRespLoss < 0.) StarchRespLoss = StarchRespL(CNamesCnt3)
   Alpha_BI = AlphaBI(i)
   if (Alpha_BI<0.) Alpha_BI= AlphaBI(CNamesCnt3)
   Beta_BI  = BetaBI(i)
   if (Alpha_BI<0.) Alpha_BI= AlphaBI(CNamesCnt3)
   Call getsrp(xMod, 'DamageIndex_Harvest', DamageIndex_Harvest)
   RDamageLossPerc = (Alpha_BI * DamageIndex_Harvest**2 + &
      Beta_BI * DamageIndex_Harvest)
   Call getsi  (xMod, 'stocktype'   , StockType)
   Call getsrp (xMod, 'stockdur'    , StockDur)
   Call getsrp (xMod, 'AlphaST'     , AlphaST)
   Call getsrp (xMod, 'stocktempb'  , StockTempB)
   StockTempSum = 0.
   DurStock = 0
   AvRespLossPerc          = 0.
   DamageLossPerc          = 0.
   UnavUBGRespLossPerc     = 0.
   UnavDMRespLossPerc      = 0.
   UnavStarchRespLossPerc  = 0.
   UBGStockLossPerc        = 0.
   DMStockLossPerc         = 0.
   StarchStockLossPerc     = 0.

   UBGCalc          = 0.
   UBGStock         = 0.
```

```
      UBGUnavResp      = 0.
      UBGAvResp        = 0.
      StarchCalc       = 0.
      StarchStock      = 0.
      StarchUnavResp   = 0.
      StarchAvResp     = 0.
      TuberWt          = 0.
      TuberWtStock     = 0.
      TuberWtUnavResp  = 0.
      TuberWtAvResp    = 0.
   else if (xNewTask == 'do_rates') then
      call getsrt (xMod, 'doy', doy)
      DamageLossPerc         = 0.
      UnavUBGRespLossPerc    = 0.
      UnavDMRespLossPerc     = 0.
      UnavStarchRespLossPerc = 0.
      AvRespLossPerc         = 0.
      UBGStockLossPerc        = 0.
      DMStockLossPerc         = 0.
      StarchStockLossPerc     = 0.
      if ((StockType == 0) .or. (StockType == 1)) then
         call getsrt (xMod,'TMDA',StockTemp)
         StockTemp = StockTemp + AlphaST
      Elseif (StockType == 2) then
         call getsrt (xMod,'TMDAN',StockTemp)
         StockTemp = StockTemp + AlphaST
      elseif (StockType==3) then
         StockTemp = AlphaST
      else
         call fatalerr(xMod, 'stocktype not found; should be value from 0 to 3')
      end if
   else if (xNewTask == 'do_states') then
      call getsi (xMod, 'DateCmp3', DateCmp3)
      if (DateCmp3 <= 0) then
         call getsrt(xMod, 'TuberWt', TuberWt)
         call getsrt(xMod, 'UBGCalc', UBGCalc)
         call getsrt(xMod, 'StarchCalc', StarchCalc)
         UBGStock         = UBGCalc
         UBGUnavResp      = UBGCalc
         UBGAvResp        = UBGCalc
         StarchStock      = StarchCalc
         StarchUnavResp   = StarchCalc
         StarchAvResp     = StarchCalc
         TuberWtStock     = TuberWt
         TuberWtUnavResp  = TuberWt
         TuberWtAvResp    = TuberWt
      elseif (DateCmp3 > 0) then
         DurStock = DurStock + 1
         if (DurStock <= 14) then
            DamageLossPerc = max(0., RDamageLossPerc * 2. * (14. - DurStock) / &
               (14.**2))
         else
            DamageLossPerc     = 0.
         end if
         StockTempSumChange = Max(0., StockTemp-StockTempB)
         StockTempSum       = StockTempSum + StockTempSumChange
         AvRespLossPerc     = RAvLoss * 100. * StockTempSumChange
         if (DurStock <= 24 * 7) then
            X = max(0., (2.* 24.*7. - 2. * DurStock) / ((24. * 7.)**2))
            UnAvUbgRespLossPerc    = UBGRespLoss * X
            UnAvDMRespLossPerc     = DMRespLoss * X
            UnAvStarchRespLossPerc = StarchRespLoss * X
         else
            UnAvUbgRespLossPerc    = 0.
            UnAvDMRespLossPerc     = 0.
            UnAvStarchRespLossPerc = 0.
         end if
         UBGStockLossPerc = AvRespLossPerc + DamageLossPerc + UnavUBGRespLossPerc
```

```
      DMStockLossPerc = AvRespLossPerc + DamageLossPerc + UnavDMRespLossPerc
      StarchStockLossPerc =AvRespLossPerc+DamageLossPerc+UnavStarchRespLossPerc
      UBGStock         = UBGStock * (1. - UBGStockLossPerc / 100.)
      UBGUnavResp      = UBGUnavResp * (1. - UnAvUBGRespLossPerc / 100.)
      UBGAvResp        = UBGAvResp * (1. -  AvRespLossPerc / 100.)
      StarchStock      = StarchStock * (1. - StarchStockLossPerc / 100.)
      StarchUnavResp   = StarchUnavResp * (1. - UnAvStarchRespLossPerc / 100.)
      StarchAvResp     = StarchAvResp * (1. - AvRespLossPerc / 100.)
      TuberWtStock     = TuberWtStock * (1. - DMStockLossPerc / 100.)
      TuberWtUnavResp  = TuberWtUnavResp * (1. - UnAvDMRespLossPerc / 100.)
      TuberWtAvResp    = TuberWtAvResp * (1. - AvRespLossPerc / 100.)
   endif
else if (xNewTask == 'output') then
   call outdat(2,0, 'UBGStock',        UBGStock)
   call outdat(2,0, 'UBGUnAvResp',     UBGUnAvResp)
   call outdat(2,0, 'UBGAvResp',       UBGAvResp)
   call outdat(2,0, 'UBGStorPercLoss', &
               100. * (UBGCalc - UBGStock) / max(1.e-10, UBGCalc))
   call outdat(2,0, 'UBGAvRespStorPercLoss', &
               100. * (UBGCalc - UBGAvResp) / max(1.e-10, UBGCalc))
   call outdat(2,0, 'UBGUnAvRespStorPercLoss', &
               100. * (UBGCalc - UBGUnAvResp) / max(1.e-10, UBGCalc))
   call outdat(2,0, 'StarchStock',     StarchStock)
   call outdat(2,0, 'StarchUnAvResp',  StarchUnAvResp)
   call outdat(2,0, 'StarchAvResp',    StarchAvResp)
   call outdat(2,0, 'StarchStorPercLoss', &
               100. * (StarchCalc - StarchStock) / max(1.e-10, StarchCalc))
   call outdat(2,0, 'StarchAvRespStorPercLoss', &
               100. * (StarchCalc - StarchAvResp) / max(1.e-10, StarchCalc))
   call outdat(2,0, 'StarchUnAvRespStorPercLoss', &
               100. * (StarchCalc - StarchUnAvResp) / max(1.e-10, StarchCalc))
   call outdat(2,0, 'TuberWtStock',    TuberWtStock)
   call outdat(2,0, 'TuberWtUnAvResp', TuberWtUnAvResp)
   call outdat(2,0, 'TuberWtAvResp',   TuberWtAvResp)
   call outdat(2,0, 'TuberWtStorPercLoss', &
               100. * (TuberWt - TuberWtStock) / max(1.e-10, TuberWt))
   call outdat(2,0, 'TuberWtAvRespStorPercLoss', &
               100. * (TuberWt - TuberWtAvResp) / max(1.e-10, TuberWt))
   call outdat(2,0, 'TuberWtUnAvRespStorPercLoss', &
               100. * (TuberWt - TuberWtUnAvResp) / max(1.e-10, TuberWt))
end if
Return
End subroutine stockdamage_dynamic
```

I.27. Subroutine TOTALSTRESSRATES

```
Subroutine TotalStressRates (xNewTask,xMod,xLogFileUnit)
implicit none
include '..\GetCropData\CropDataSet.inc'
include '..\GetSoilData\SoilDataSet.inc'

character*(*) :: xNewTask,xMod
character(len=13) :: OldTask, EmptyString

integer ::       xLogFileUnit
integer :: i

logical :: RootGrowthFlag, DoWaterStress, DoNitroStress, DoBlightStress,
Dosoilblight

real :: RootDepthSoilMx, RootDepthChangeP, RootDepthChange, RootHeightChangeP,&
    RootHeightChange, StressIndexLUE,  StressIndexLEff, StressIndexLvWt, &
   StressIndexRtWt, StressIndexStWt, StressIndexTuWt, StressIndexLvAr, &
   StressIndexRtDp, StressIndexRtHt, StressIndexTmSu, StressIndexSLA, &
   StressIndexStAg, StressIndexRoAg, StressIndexTuAg, StressIndexTuGr, &
   WaterStressIndexLUE, WaterStressIndexLEff, WaterStressIndexLvWt, &
   WaterStressIndexRtWt, WaterStressIndexStWt, WaterStressIndexTuWt, &
   WaterStressIndexLvAr, WaterStressIndexRtDp, WaterStressIndexRtHt, &
   WaterStressIndexTmSu, WaterStressIndexLvAg, WaterStressIndexSLA, &
   WaterStressIndexStAg, WaterStressIndexRoAg, WaterStressIndexTuAg, &
   TransPot,TransAct, WaterStressIndex, WaterStressIndexTuGr, &
   NitroStressIndexLUE,  NitroStressIndexLEff, NitroStressIndexLvWt, &
   NitroStressIndexRtWt, NitroStressIndexStWt, NitroStressIndexTuWt, &
   NitroStressIndexLvAr, NitroStressIndexRtDp, NitroStressIndexRtHt, &
   NitroStressIndexTmSu, NitroStressIndexSLA, NitroStressIndexStAg, &
   NitroStressIndexRoAg, NitroStressIndexTuAg, NitroStressIndexTuGr, &
   NitroStressIndexLvAgCl(LeafClMxn), Nuptr, Nuptrt, NuptMin, nsolconc, &
   nstabconc,photcap, AllowedAboveGrowthNlim, AllowedTuberGrowthNlim, &
   BlightStressIndexLUE,  BlightStressIndexLEff, BlightStressIndexLvWt, &
   BlightStressIndexRtWt, BlightStressIndexStWt, BlightStressIndexTuWt, &
   BlightStressIndexLvAr, BlightStressIndexRtDp, BlightStressIndexRtHt, &
   BlightStressIndexTmSu, BlightStressIndexLvAg, BlightStressIndexSLA, &
   BlightStressIndexStAg, BlightStressIndexRoAg, BlightStressIndexTuAg, &
   BlightStressIndexTuGr, LeafDis, LeafLive, LvAgeStressIndexLUE, TestPar, &
   blightgamma, blightbeta, blightdelta, blighteps, alphaWater, limit
real, parameter :: tiny = 1.e-10
save
if (xNewTask == 'initialize') then
   call getsl (xMod, 'DoWaterStress' , DoWaterStress )
   call getsl (xMod, 'DoNitroStress' , DoNitroStress )
   call getsl (xMod, 'DoBlightStress', DoBlightStress)
   call getsl (xMod, 'Dosoilblight'  , Dosoilblight)
   call Logmessage (xLogFileUnit,9,xMod,'3.0')
   if ((DoWaterStress) .or. (DoNitrostress)) then
      call getsrp (xMod, 'zrtms', RootDepthSoilMx)
   end if
   if (DoWaterStress) then
      call getsrp (xMod, 'alphaWater', alphaWater)
      call getsrp (xMod, 'KstrsAr', KstrsAr)
      call getsrp (xMod, 'BetaStrsAr', BetaStrsAr)
      call getsrp (xMod, 'KstrsAge', KstrsAge)
      call getsrp (xMod, 'Bage', Bage)
      call getsrp (xMod, 'Gage', Gage)
      call getsrp (xMod, 'TestPar', TestPar)
   endif
   if (DoBlightStress .or. DoSoilBlight) then
      call getsrp (xMod, 'BlightGamma',BlightGamma)
      call getsrp (xMod, 'BlightBeta', BlightBeta)
      call getsrp (xMod, 'BlightDelta',BlightDelta)
      call getsrp (xMod, 'BlightEps',  BlightEps)
   endif
```

```
      StressIndexLUE  = 1.
      StressIndexLEff = 1.
      StressIndexLvWt = 1.
      StressIndexLvAr = 1.
      StressIndexStWt = 1.
      StressIndexTuWt = 1.
      StressIndexRtWt = 1.
      StressIndexTmSu = 1.
      StressIndexStAg = 1.
      StressIndexRoAg = 1.
      StressIndexTuAg = 1.
      StressIndexRtDp = 1.
      StressIndexRtHt = 1.
      StressIndexSLA  = 1.
      StressIndexTuGr = 1.
   else if (xNewTask == 'do_rates') then
      call getsim (xMod, 'leafclcnt', leafclcnt, 0)
      WaterStressIndexLUE  = 1.
      WaterStressIndexLEff = 1.
      WaterStressIndexLvWt = 1.
      WaterStressIndexLvAr = 1.
      WaterStressIndexStWt = 1.
      WaterStressIndexTuWt = 1.
      WaterStressIndexRtWt = 1.
      WaterStressIndexTmSu = 1.
      WaterStressIndexLvAg = 1.
      WaterStressIndexStAg = 1.
      WaterStressIndexRoAg = 1.
      WaterStressIndexTuAg = 1.
      WaterStressIndexTuGr = 1.
      WaterStressIndexRtDp = 1.
      WaterStressIndexRtHt = 1.
      WaterStressIndexSLA  = 1.
      NitroStressIndexLUE  = 1.
      NitroStressIndexLEff = 1.
      NitroStressIndexLvWt = 1.
      NitroStressIndexLvAr = 1.
      NitroStressIndexStWt = 1.
      NitroStressIndexTuWt = 1.
      NitroStressIndexRtWt = 1.
      NitroStressIndexTmSu = 1.
      NitroStressIndexStAg = 1.
      NitroStressIndexRoAg = 1.
      NitroStressIndexTuAg = 1.
      NitroStressIndexTuGr = 1.
      NitroStressIndexRtDp = 1.
      NitroStressIndexRtHt = 1.
      NitroStressIndexSLA  = 1.
      if (leafclcnt > 0) then
         do i = 1, leafclcnt
            NitroStressIndexLvAgCl(i) =  1.
         end do
      endif
      BlightStressIndexLUE  = 1.
      BlightStressIndexLEff = 1.
      BlightStressIndexLvWt = 1.
      BlightStressIndexLvAr = 1.
      BlightStressIndexStWt = 1.
      BlightStressIndexTuWt = 1.
      BlightStressIndexRtWt = 1.
      BlightStressIndexTmSu = 1.
      BlightStressIndexLvAg = 1.
      BlightStressIndexStAg = 1.
      BlightStressIndexRoAg = 1.
      BlightStressIndexTuAg = 1.
      BlightStressIndexTuGr = 1.
      BlightStressIndexRtDp = 1.
      BlightStressIndexRtHt = 1.
```

```
      BlightStressIndexSLA  = 1.
      LvAgeStressIndexLue = 1.
      if (LeafClCnt > 0 ) then
         if (DoWaterStress) then
            call getsrt (xMod, 'RootDepth', RootDepth)
            call getsrt (xMod, 'RootHeight', RootHeight)
            call getsrt (xMod, 'RootDepthChangeP', RootDepthChangeP)
            call getsrt (xMod, 'RootHeightChangeP', RootHeightChangeP)
            RootDepthChange = 0.
            RootHeightChange = 0.
            WaterStressIndexRtDp = 0.
            WaterStressIndexRtHt = 0.
!           roots can grow, avoid that they get too deep
            RootDepthChange = max (RootDepthSoilMx-RootDepth,0.)
            RootHeightChange = min (ULimit(1) - RootHeight,0.)
            if (RootDepthChangeP > 0.) then
               WaterStressIndexRtDp = min (1., max (0., RootDepthChange / &
                  (RootDepthChangeP + 1.e-10)))
            else
               WaterStressIndexRtDp = 0.
            end if
            if (RootHeightChange < 0.) then
               WaterStressIndexRtHt = min (1., max (0., RootHeightChange / &
                  (RootHeightChangeP + 1.e-10)))
            else
               WaterStressIndexRtHt = 0.
            end if
            call getsrt (xMod,'TransPot',TransPot)
            call getsrt (xMod,'TransAct',TransAct)
            if (TransPot > 0.) then
               WaterStressIndex     = limit (0.,1.,(TransAct/TransPot))
               WaterStressIndexLUE  = (WaterStressIndex *(1. + alphaWater)) / &
                  (WaterStressIndex + alphaWater)
               WaterStressIndexLEff = 1.
               WaterStressIndexLvWt = 1.
               WaterStressIndexLvAr = 1. + 1. /(1.+exp(max(-30.,min(30., - &
                  KstrsAr*(WaterStressIndex - BetastrsAr))))) - &
                  1. /(1.+exp(max(-30.,min(30., -KstrsAr*(1. - BetastrsAr)))))
               WaterStressIndexStWt = 1.
               WaterStressIndexTuWt = 1.
               WaterStressIndexRtWt = 1.
               WaterStressIndexTmSu = 1.
               waterstressindexLvAg = 1. + Gage * (exp(max(-30., min(30.,&
                  -kstrsage * (WaterStressIndex - bage)))) - &
                  exp(max(-30., min(30., -kstrsage * (1. - bage)))))
            end if
         end if
         if (DoNitroStress) then
            call getsrt (xMod, 'AllowedAboveGrowthNLim', AllowedAboveGrowthNlim)
            call getsrt (xMod, 'AllowedTuberGrowthNLim', AllowedTuberGrowthNlim)
            call getsrt (xMod, 'NuptR',  NuptR)
             NitroStressIndexLvAr = AllowedAboveGrowthNLim
             NitroStressIndexTuGr = AllowedTuberGrowthNLim
         end if
      end if
      StressIndexLUE  = min ( WaterStressIndexLUE , NitroStressIndexLUE, &
          LvAgeStressIndexLUE ) * BlightStressIndexLue
      StressIndexLEff = min ( WaterStressIndexLEff, NitroStressIndexLEff) * &
          BlightStressIndexLEff
      StressIndexLvAr = min ( WaterStressIndexLvAr, NitroStressIndexLvAr) * &
          BlightStressIndexLvAr
      StressIndexLvWt = min ( WaterStressIndexLvWt, NitroStressIndexLvWt) * &
          BlightStressIndexLvWt
      StressIndexStWt = min ( WaterStressIndexStWt, NitroStressIndexStWt) * &
          BlightStressIndexStWt
      StressIndexTuWt = min ( WaterStressIndexTuWt, NitroStressIndexTuWt) * &
          BlightStressIndexTuWt
      StressIndexRtWt = min ( WaterStressIndexRtWt, NitroStressIndexRtWt) * &
```

```
202          BlightStressIndexRtWt
203       StressIndexTmSu = min ( WaterStressIndexTmSu, NitroStressIndexTmSu) * &
204          BlightStressIndexTmSu
205       StressIndexStAg = 1. - min ( WaterStressIndexStAg, NitroStressIndexStAg) * &
206          BlightStressIndexStAg
207       StressIndexRoAg = 1. - min ( WaterStressIndexRoAg, NitroStressIndexRoAg) * &
208          BlightStressIndexRoAg
209       StressIndexTuAg = 1. - min ( WaterStressIndexTuAg, NitroStressIndexTuAg) * &
210          BlightStressIndexTuAg
211       StressIndexTuGr = min ( WaterStressIndexTuGr, NitroStressIndexTuGr) * &
212          BlightStressIndexTuGr
213       StressIndexRtDp = min ( WaterStressIndexRtDp, NitroStressIndexRtDp) * &
214          BlightStressIndexRtDp
215       StressIndexRtHt = min ( WaterStressIndexRtHt, NitroStressIndexRtHt) * &
216          BlightStressIndexRtHt
217       StressIndexSLA  = min ( WaterStressIndexSLA,  NitroStressIndexSLA)  * &
218          BlightStressIndexSLA
219       if (leafclcnt > 0) then
220          do i = 1, leafclcnt
221             StressIndexLvAgCl(i) = max( WaterStressIndexLvAg, &
222                 NitroStressIndexLvAgCl(i)) * BlightStressIndexLvAg
223          end do
224       endif
225       call cmdelvar ('StressIndexLUE')
226       call cmdelvar ('StressIndexLEff')
227       call cmdelvar ('StressIndexLvAr')
228       call cmdelvar ('StressIndexLvAg')
229       call cmdelvar ('StressIndexStAg')
230       call cmdelvar ('StressIndexRoAg')
231       call cmdelvar ('StressIndexTuAg')
232       call cmdelvar ('StressIndexTuGr')
233       call cmdelvar ('StressIndexLvAgCl')
234       call cmdelvar ('StressIndexLvWt')
235       call cmdelvar ('StressIndexStWt')
236       call cmdelvar ('StressIndexTuWt')
237       call cmdelvar ('StressIndexRtWt')
238       call cmdelvar ('StressIndexRtDp')
239       call cmdelvar ('StressIndexRtHt')
240       call cmdelvar ('StressIndexTmSu')
241       call cmdelvar ('StressIndexSLA')
242       call putsrt (xMod, 'StressIndexLUE' , StressIndexLUE )
243       call putsrt (xMod, 'StressIndexLEff', StressIndexLEff)
244       call putsrt (xMod, 'StressIndexLvAr', StressIndexLvAr)
245       call putsrt (xMod, 'StressIndexStAg', StressIndexStAg)
246       call putsrt (xMod, 'StressIndexRoAg', StressIndexRoAg)
247       call putsrt (xMod, 'StressIndexTuAg', StressIndexTuAg)
248       call putsrt (xMod, 'StressIndexTuGr', StressIndexTuGr)
249       call putsrt (xMod, 'StressIndexLvWt', StressIndexLvWt)
250       call putsrt (xMod, 'StressIndexStWt', StressIndexStWt)
251       call putsrt (xMod, 'StressIndexTuWt', StressIndexTuWt)
252       call putsrt (xMod, 'StressIndexRtWt', StressIndexRtWt)
253       call putsrt (xMod, 'StressIndexRtDp', StressIndexRtDp)
254       call putsrt (xMod, 'StressIndexRtHt', StressIndexRtHt)
255       call putsrt (xMod, 'StressIndexTmSu', StressIndexTmSu)
256       call putsrt (xMod, 'StressIndexSLA' , StressIndexSLA )
257    else if (xNewTask == 'output') then
258       WaterStressIndexLvAr = 1. + 1. /(1.+exp(max(-30.,min(30., -KstrsAr * &
259          (WaterStressIndex - BetastrsAr))))) - 1./(1.+exp(max(-30.,min(30., &
260           -KstrsAr*(1. - BetastrsAr)))))
261    end if
262    end subroutine TotalStressRates
```

I.28. Subroutine TOTALSTRESSSTATES

```
Subroutine TotalStressStates (xNewTask,xMod,xLogFileUnit)
implicit none
include '..\GetCropData\CropDataSet.inc'
include '..\GetSoilData\SoilDataSet.inc'

character*(*) :: xNewTask,xMod
character*30 :: FINTtext
character(len=13) :: OldTask, EmptyString
integer :: xLogFileUnit, i, tmpi1, FintExt
logical :: RootGrowthFlag, DoWaterStress, DoNitroStress, DoBlightStress, &
           Dosoilblight

real :: RootDepthChangeP, RootDepthChange, LaiDis, FrDisLai, LeafArN, &
   StressIndexLUE,  StressIndexLEff, StressIndexECPDF, StressIndexPhot, &
   StressIndexLvWt, StressIndexRtWt, StressIndexStWt, StressIndexTuWt, &
   StressIndexLvAr, StressIndexRtDp, StressIndexRtHt, StressIndexTmSu, &
   StressindexPAMAX, StressIndexSLA,  StressIndexStAg, StressIndexRoAg, &
   StressIndexTuAg, WaterStressIndexLUE, WaterStressIndexLEff, &
   WaterStressIndexECPDF, WaterStressIndexLvWt, WaterStressIndexRtWt, &
   WaterStressIndexStWt, WaterStressIndexTuWt, WaterStressIndexLvAr, &
   WaterStressIndexRtDp, WaterStressIndexRtHt, WaterStressIndexTmSu, &
   WaterStressindexPAMAX, WaterStressIndexPhot, WaterStressIndexSLA, &
   WaterStressIndexStAg, WaterStressIndexRoAg, WaterStressIndexTuAg, &
   TransPot,TransAct, WaterStressIndex, NitroStressIndexLUE, &
   NitroStressIndexLEff, NitroStressIndexECPDF, NitroStressIndexLvWt, &
   NitroStressIndexRtWt, NitroStressIndexStWt, NitroStressIndexTuWt, &
   NitroStressIndexLvAr, NitroStressIndexRtDp, NitroStressIndexRtHt, &
   NitroStressIndexTmSu, NitroStressindexPAMAX, NitroStressIndexPhot, &
   NitroStressIndexSLA,  NitroStressIndexStAg, NitroStressIndexRoAg, &
   NitroStressIndexTuAg, NitroStressIndexLvAgCl(LeafClMxn), LeafTotCl, NdemT, &
   Nuptr, Ndemtt, Nuptrt, nsolconc, nstabconcLeafphot, ntotconcleafwtphot, &
   NStabConcStem, NStabConcStemVol, photcap, NStabLeafLive, NStabLeafMax, &
   NStabLeafMin, NstrucLeafMin, ntotconctotal, FrParAbsAbove, FrParAbs, &
   FrParAbsL, FrParAbsT, BNRGRL, NStabStemMax, DMNStabLeaf, DMNStrucLeaf, &
   DMNStabStem, DMNStrucStem, DMNStabRoot, DMNStrucRoot, DMNStabTuber, &
   DMNStrucTuber, sumtrans, partransfraction(leafclmxn), NStabConc, &
   LeafWtClTot, BlightStressIndexLUE,  BlightStressIndexLEff, &
   BlightStressIndexECPDF, BlightStressIndexLvWt, BlightStressIndexRtWt, &
   BlightStressIndexPhot, BlightStressIndexStWt, BlightStressIndexTuWt, &
   BlightStressIndexLvAr, BlightStressIndexRtDp, BlightStressIndexRtHt, &
   BlightStressIndexTmSu, BlightStressIndexLvAg, BlightStressindexPAMAX, &
   BlightStressIndexSLA,  BlightStressIndexStAg, BlightStressIndexRoAg, &
   BlightStressIndexTuAg, LeafDis, LeafLive, ECPDFAv,  LvAgeStressIndexLUE, &
   TestPar, KNLvAg, BNLvAg, KNStAg, BNStAg, alphaNSLA, betaNSLA, alphaPAMAX, &
   betaPAMAX, NStabLeafClConc, NtotConcLeafCl, NtotConcLeaf, alphaPhot, &
   betaPhot, gammaPhot, blightgamma, blightbeta, blightdelta, blighteps, &
   limit, FuncECPDF
real, parameter :: tiny = 1.e-10
save
if (xNewTask == 'initialize') then
   call getsl (xMod, 'DoWaterStress' , DoWaterStress )
   call getsl (xMod, 'DoNitroStress' , DoNitroStress )
   call getsl (xMod, 'DoBlightStress', DoBlightStress)
   call getsl (xMod, 'Dosoilblight'  , Dosoilblight)
   call Logmessage (xLogFileUnit,9,xMod,'3.0')
   call getsrp (xMod, 'ecpdf', ecpdf)
   call getsi (xMod, 'FintExt' , FintExt)
   if ((DoWaterStress) .or. (DoNitroStress)) then
      call getsi (xMod, 'nl', nl)
      call getarp (xMod, 'wcwu', wcwu, Nl, tmpi1)
      call getarp (xMod, 'wcwp', wcwp, Nl, tmpi1)
      call getarp (xMod, 'wcwo', wcwo, Nl, tmpi1)
   endif
   if (DoNitroStress) then
      call getsrp (xMod, 'knlvag',KNLvAg)
```

```
66          call getsrp (xMod, 'Bnlvag',BNLvAg)
67          call getsrp (xMod, 'knStag',KNStAg)
68          call getsrp (xMod, 'BnStag',BNStAg)
69          call getsrp (xMod, 'KNLUE',KNLUE)
70          call getsrp (xMod, 'Noffset',Noffset)
71          call getsrp (xMod, 'NStabLeafMax',NStabLeafMax)
72          call getsrp (xMod, 'NStabLeafMin',NStabLeafMin)
73          call getsrp (xMod, 'NStabStemMax',NStabStemMax)
74          call getsrp (xMod, 'NStrucLeafMin',NStrucLeafMin)
75          NStabLeafMin  = min(NStabLeafMin,  NStabLeafMax)
76          call getsrp (xMod, 'alphaPAMAX', alphaPAMAX)
77          call getsrp (xMod, 'betaPAMAX', betaPAMAX)
78          call getsrp (xMod, 'alphaPhot', alphaPhot)
79          call getsrp (xMod, 'betaPhot', betaPhot)
80          call getsrp (xMod, 'gammaPhot', gammaPhot)
81          call getsrp (xMod, 'KNRGRL',KNRGRL)
82          call getsrp (xMod, 'BNRGRL',BNRGRL)
83       else
84          call getsrp(xMod, 'TMSumLeafGrowth', TMSumLeafGrowth)
85          call getsrp(xMod, 'leafpar', leafpar)
86       endif
87       if (DoBlightStress .or. DoSoilBlight) then
88          call getsrp (xMod, 'BlightGamma',BlightGamma)
89          call getsrp (xMod, 'BlightBeta', BlightBeta)
90          call getsrp (xMod, 'BlightDelta',BlightDelta)
91          call getsrp (xMod, 'BlightEps',  BlightEps)
92       endif
93       do i = 1, leafclmxn
94          DistrDMLeafCl(i) = 0.
95       end do
96       nuptrt = 0.
97       ndemtt = 0.
98       StressIndexLUE   = 1.
99       StressIndexPAMAX = 1.
100      StressIndexLEff = 1.
101      StressIndexLvWt = 1.
102      StressIndexLvAr = 1.
103      StressIndexStWt = 1.
104      StressIndexTuWt = 1.
105      StressIndexRtWt = 1.
106      StressIndexTmSu = 1.
107      StressIndexStAg = 1.
108      StressIndexRoAg = 1.
109      StressIndexTuAg = 1.
110      StressIndexRtDp = 1.
111      StressIndexRtHt = 1.
112      StressIndexSLA  = 1.
113      StressIndexPhot = 1.
114   else if (xNewTask == 'do_rates') then
115      call getsim (xMod, 'leafclcnt', leafclcnt, 0)
116      if ((DoNitroStress) .and. (leafclcnt > 0)) then
117         call getsrt (xMod, 'Nsolconc', Nsolconc)
118         call getsrt (xMod, 'NstabconcleafphotNstabconcleafPhot)
119         call getsrt (xMod, 'NtotconcleafwtphotNtotconcleafwtPhot)
120         call getsrt (xMod, 'NStabConcStemVol',NStabConcStemVol)
121         call getsrt (xMod, 'Ntotconcleaf',Ntotconcleaf)
122         call getsrt (xMod, 'Ntotconctotal',Ntotconctotal)
123         call getsrt (xMod, 'NStabLeafLive', NStabLeafLive)
124      end if
125      WaterStressIndexLUE  = 1.
126      WaterStressIndexPAMAX = 1.
127      WaterStressIndexECPDF = 1.
128      WaterStressIndexLEff = 1.
129      WaterStressIndexLvWt = 1.
130      WaterStressIndexLvAr = 1.
131      WaterStressIndexStWt = 1.
132      WaterStressIndexTuWt = 1.
133      WaterStressIndexRtWt = 1.
```

```
      WaterStressIndexTmSu = 1.
      WaterStressIndexStAg = 1.
      WaterStressIndexRoAg = 1.
      WaterStressIndexTuAg = 1.
      WaterStressIndexRtDp = 1.
      WaterStressIndexRtHt = 1.
      WaterStressIndexSLA  = 1.
      WaterStressIndexPhot  = 1.
      NitroStressIndexLUE  = 1.
      NitroStressIndexPAMAX = 1.
      NitroStressIndexECPDF = 1.
      NitroStressIndexLEff = 1.
      NitroStressIndexLvWt = 1.
      NitroStressIndexLvAr = 1.
      NitroStressIndexStWt = 1.
      NitroStressIndexTuWt = 1.
      NitroStressIndexRtWt = 1.
      NitroStressIndexTmSu = 1.
      NitroStressIndexStAg = 1.
      NitroStressIndexRoAg = 1.
      NitroStressIndexTuAg = 1.
      NitroStressIndexRtDp = 1.
      NitroStressIndexRtHt = 1.
      NitroStressIndexSLA  = 1.
      NitroStressIndexPhot  = 1.
      if (leafclcnt > 0) then
         LeafTotCl = 0.
         do i = 1, leafclcnt
            NitroStressIndexLvAgCl(i) =  1.
            LeafTotCl = LeafTotCl + LeafArLiveCl(i)
         end do
         do i = 1, leafclcnt
             DistrDMLeafCl(i) = LeafArLiveCl(i)/max(1.e-10, LeafTotCl)
         end do
      endif

      BlightStressIndexLUE  = 1.
      BlightStressIndexPAMAX = 1.
      BlightStressIndexECPDF = 1.
      BlightStressIndexLEff = 1.
      BlightStressIndexLvWt = 1.
      BlightStressIndexLvAr = 1.
      BlightStressIndexStWt = 1.
      BlightStressIndexTuWt = 1.
      BlightStressIndexRtWt = 1.
      BlightStressIndexTmSu = 1.
      BlightStressIndexStAg = 1.
      BlightStressIndexRoAg = 1.
      BlightStressIndexTuAg = 1.
      BlightStressIndexRtDp = 1.
      BlightStressIndexRtHt = 1.
      BlightStressIndexSLA  = 1.
      BlightStressIndexPhot = 1.
      LvAgeStressIndexLue = 1.
      if ((DoWaterStress .or. DoNitroStress) .and. (leafclcnt > 0)) then
         call getsrt (xMod, 'RootDepth', RootDepth)
         call getsrt (xMod, 'RootHeight', RootHeight)
         WaterStressIndexRtDp = 1.
         WaterStressIndexRtHt = 1.
         do i= 3, Nl
            if ((RootDepth > ULimit(i)) .and. (RootDepth <= LLimit(i)))  then
               if (wclqt(i) < wcwu(i)) then
                  WaterStressIndexRtDp = min(1., max(0., (wclqt(i) - wcwp(i)) / &
                     max(1.e-6, wcwu(i) - wcwp(i))))
               elseif (wclqt(i) > wcwo(i)) then
                  WaterStressIndexRtDp = min(1., max(0., (wcst(i) - wclqt(i)) / &
                     max(1.e-6, wcst(i) - wcwo(i))))
               else
```

```
                WaterStressIndexRtDp = 1.
             end if
          end if
       end do
       do i= 1, 2
          if ((RootHeight > ULimit(i)) .and. (RootHeight <= LLimit(i)))  then
             if (wclqt(i) < wcwu(i)) then
                WaterStressIndexRtHt = min(1., max(0., (wclqt(i) - wcwp(i)) / &
                   max(1.e-6, wcwu(i) - wcwp(i))))
             elseif (wclqt(i) > wcwo(i)) then
                WaterStressIndexRtHt = min(1., max(0., (wcst(i) - wclqt(i)) / &
                   max(1.e-6, wcst(i) - wcwo(i))))
             else
                WaterStressIndexRtHt = 1.
             end if
          end if
       end do
    endif
    if ((DoNitroStress) .and. (leafclcnt > 0)) then
       NitroStressIndexLvWt = 1.
       NitroStressIndexLvAr = max(0., 1. - exp(max(-30.,min(30., -KNRGRL * &
          (NSolConc - BNRGRL)))))
       NitroStressIndexStWt = 1.
       NitroStressIndexTuWt = 1.
       NitroStressIndexRtWt = 1.
       NitroStressIndexTmSu = 1.
       LeafAr = 0.
       SumTrans = 0.
       do i = LeafClCnt, 1, -1
          if (i < LeafClCnt) LeafAr = LeafAr + LeafArLiveCl(i+1)/10000.
          Partransfraction(i) = LeafArLiveCl(i) * exp (max(-30.,min(30., &
             -ECPDFAv * LeafAr)))
          sumtrans = sumtrans + partransfraction(i)
       end do
       if (sumtrans > 0.) then
          NitroStressIndexLUE = 0.
          do i = 1, LeafClCnt
             if (LeafAliveCl(i)) then
                NStabConc = NStabLiveCl(i)/max(1.e-8,LeafArLiveCl(i)/10000.)
                NitroStressIndexLUE = NitroStressIndexLUE + &
                   (partransfraction(i) / sumtrans) * max(0.,1. - exp(max( &
                   -30.,min(30.,-KNLUE * (NStabConc - NOffset)/max(1.e-6, &
                   NStabLeafMax - Noffset)))))
             end if
          end do
       else
          NitroStressIndexLUE = 1.
       end if
       NitrostressindexPAMAX = max(0.,1. - exp(max(-30.,min(30.,-alphaPAMAX &
          * (NTotconcLeafwtPhot - betaPAMAX)))))
       NitrostressindexPhot  = max(0.,min(1., alphaPhot * 6.75 * (0.1 * &
           NStabconcLeafPhot - betaPhot) / (alphaPhot * 6.75 * (0.1 * &
           NStabconcLeafPhot - betaPhot) + gammaPhot)))
       NitroStressIndexLEff = 1.
       NitroStressIndexStAg = max(0., min(1., 1. - exp(max(-30.,min(30., &
          -KNStAg * (NStabConcStemVol/max(1.e-10,NStabStemMax)-BNStag))))))
       FrParAbsAbove = 0.
       LeafArN = 0.
       NitroStressIndexECPDF = 1.
       do i = LeafClCnt, 1, -1
          NtotConcLeafCl = (NStabLiveCl(i) + NStrucLiveCl(i) + NSolLiveCl(i)) /&
             max(1.e-10, LeafArLiveCl(i)/10000.)
          NitroStressIndexLvAgCl(i) = 1. + exp(max(-30., min(30., -KNLvAg * &
             (NtotConcLeafCl-BNLvAg))))
       end do
       NitroStressIndexRtDp = 1.
    elseif((LeafClCnt > 0) .and. not(DoNitroStress) ) then
       NitroStressIndexECPDF = 1.
```

```
      call getsrt(xMod, 'TMSum', TMSum)
      if (TMSum < TMSumLeafGrowth) then
         nitrostressindexlvar = max(0.,(1.-(TMSum/max(1.e-6, &
            TMSumLeafGrowth))))**LeafPar
      else
         nitrostressindexlvar = 0.
      end if
   end if
   FrDisLai = 0.
   if (DoBlightStress .or. DoSoilBlight) then
      call getsrt (xMod,'LaiLive', LaiLive )
      call getsrt (xMod,'LaiDis', LaiDis)
      frdislai = LaiDis / max(tiny, LaiLive)
      BlightStressIndexLUE  = blightgamma*(max(0.,min(1.,(1.-&
         frdislai)))**blightbeta)
      BlightStressIndexLEff = 1.
      BlightStressIndexLvWt = 1.
      BlightStressIndexLvAr = 1.
      BlightStressIndexStWt = 1.
      BlightStressIndexTuWt = 1.
      BlightStressIndexRtWt = 1.
      BlightStressIndexTmSu = 1.
      BlightStressIndexStAg = 1.
      BlightStressIndexLvAg = 1. + blightdelta * max(0.,(1. - exp(max(-30., &
         min(30., -blighteps*frdislai)))))
      BlightStressIndexRtDp = 1.
   end if
   StressIndexLUE  = min ( WaterStressIndexLUE , NitroStressIndexLUE, &
      LvAgeStressIndexLUE ) * BlightStressIndexLue
   StressIndexPAMAX= min ( WaterStressIndexPAMAX, NitroStressIndexPAMAX, &
      LvAgeStressIndexLUE ) * BlightStressIndexPAMAX
   StressIndexECPDF= min ( WaterStressIndexECPDF, NitroStressIndexECPDF ) * &
      BlightStressIndexECPDF
   StressIndexLEff = min ( WaterStressIndexLEff, NitroStressIndexLEff) * &
      BlightStressIndexLEff
   StressIndexLvAr = min ( WaterStressIndexLvAr, NitroStressIndexLvAr) * &
      BlightStressIndexLvAr
   StressIndexLvWt = min ( WaterStressIndexLvWt, NitroStressIndexLvWt) * &
      BlightStressIndexLvWt
   StressIndexStWt = min ( WaterStressIndexStWt, NitroStressIndexStWt) * &
      BlightStressIndexStWt
   StressIndexTuWt = min ( WaterStressIndexTuWt, NitroStressIndexTuWt) * &
      BlightStressIndexTuWt
   StressIndexRtWt = min ( WaterStressIndexRtWt, NitroStressIndexRtWt) * &
      BlightStressIndexRtWt
   StressIndexTmSu = min ( WaterStressIndexTmSu, NitroStressIndexTmSu) * &
      BlightStressIndexTmSu
   StressIndexStAg = 1. - min ( WaterStressIndexStAg, NitroStressIndexStAg) * &
      BlightStressIndexStAg
   StressIndexRoAg = 1. - min ( WaterStressIndexRoAg, NitroStressIndexRoAg) * &
      BlightStressIndexRoAg
   StressIndexTuAg = 1. - min ( WaterStressIndexTuAg, NitroStressIndexTuAg) * &
      BlightStressIndexTuAg
   StressIndexRtDp = min ( WaterStressIndexRtDp, NitroStressIndexRtDp) * &
      BlightStressIndexRtDp
   StressIndexRtHt = min ( WaterStressIndexRtHt, NitroStressIndexRtHt) * &
      BlightStressIndexRtHt
   StressIndexSLA  = min ( WaterStressIndexSLA,  NitroStressIndexSLA)  * &
      BlightStressIndexSLA
   StressIndexPhot  = min ( WaterStressIndexPhot,  NitroStressIndexPhot)  * &
     BlightStressIndexPhot
   if (leafclcnt > 0) then
      do i = 1, leafclcnt
         StressIndexLvAgCl(i) = NitroStressIndexLvAgCl(i)
      end do
   endif
   call cmdelvar ('StressIndexLUE')
   call cmdelvar ('StressIndexPAMAX')
```

```
      call cmdelvar ('StressIndexECPDF')
      call cmdelvar ('StressIndexLEff')
      call cmdelvar ('StressIndexLvAr')
      call cmdelvar ('StressIndexLvAg')
      call cmdelvar ('StressIndexStAg')
      call cmdelvar ('StressIndexRoAg')
      call cmdelvar ('StressIndexTuAg')
      call cmdelvar ('StressIndexLvWt')
      call cmdelvar ('StressIndexStWt')
      call cmdelvar ('StressIndexTuWt')
      call cmdelvar ('StressIndexRtWt')
      call cmdelvar ('StressIndexRtDp')
      call cmdelvar ('StressIndexRtHt')
      call cmdelvar ('StressIndexTmSu')
      call cmdelvar ('StressIndexSLA')
      call cmdelvar ('StressIndexPhot')
      call putsrt (xMod, 'StressIndexLUE' , StressIndexLUE )
      call putsrt (xMod, 'StressIndexPAMAX' , StressIndexPAMAX )
      call putsrt (xMod, 'StressIndexECPDF' , StressIndexECPDF )
      call putsrt (xMod, 'StressIndexLEff', StressIndexLEff)
      call putsrt (xMod, 'StressIndexLvAr', StressIndexLvAr)
      call putsrt (xMod, 'StressIndexStAg', StressIndexStAg)
      call putsrt (xMod, 'StressIndexRoAg', StressIndexRoAg)
      call putsrt (xMod, 'StressIndexTuAg', StressIndexTuAg)
      call putsrt (xMod, 'StressIndexLvWt', StressIndexLvWt)
      call putsrt (xMod, 'StressIndexStWt', StressIndexStWt)
      call putsrt (xMod, 'StressIndexTuWt', StressIndexTuWt)
      call putsrt (xMod, 'StressIndexRtWt', StressIndexRtWt)
      call putsrt (xMod, 'StressIndexRtDp', StressIndexRtDp)
      call putsrt (xMod, 'StressIndexRtHt', StressIndexRtHt)
      call putsrt (xMod, 'StressIndexTmSu', StressIndexTmSu)
      call putsrt (xMod, 'StressIndexSLA' , StressIndexSLA )
      call putsrt (xMod, 'StressIndexPhot' , StressIndexPhot )
   else if (xNewTask == 'output') then
      NtotConcLeafCl = (NStabLiveCl(1) + NStrucLiveCl(1) + NSolLiveCl(1)) / &
         max(1.e-10, LeafArLiveCl(1)/10000.)
      continue
   end if
   end subroutine TotalStressStates
```

I.29. Real function FUNCECPDF

```
real Function FuncECPDF (ECPDFType, ECPDFChoice, LeafArea)
implicit none
integer :: ECPDFChoice
character*30, parameter :: xMod = 'FuncECPDF'
character*30 :: ECPDFType, FINTtext, SCANtext, SCANtextGRN, SCANtextRED
real :: LeafArea, LeafWtPresent, StemWtPresent, ecpdf, ecpdfmax, alphaecpdf, &
   kminfint, kmaxfint, betakminfint, betakmaxfint, akfint, bkfint, kminscan, &
   kmaxscan, betakminscan, betakmaxscan, akscan, bkscan, alphascan, betascan, &
   GroundCover, LaiMax
call getsc (xMod, 'FINTtext', FINTtext)
call getsc (xMod, 'SCANtext', SCANtext)
call getsc (xMod, 'SCANtextGRN', SCANtextGRN)
call getsc (xMod, 'SCANtextRED', SCANtextRED)
if (ECPDFChoice == 1) then
   call getsrp (xMod, 'ecpdf', ecpdf)
   if (ECPDFType == FINTtext) then
      call getsrtm(xMod, 'LaiMx', LaiMax, 0.)
      call getsrp (xMod, 'ecpdf', ecpdf)
      call getsrp (xMod, 'alphaecpdf', alphaecpdf)
      call getsrp (xMod, 'ecpdfmax', ecpdfmax)
      if (LeafArea < LaiMax) then
         FuncECPDF = ecpdf + (ecpdfmax-ecpdf) * exp(max(-30., min(30., &
            -alphaecpdf * leafarea)))
      else
         FuncECPDF =  ecpdf
      end if
   elseif (ECPDFType == SCANtext) then
      call getsrp (xMod, 'kmaxscan', kmaxscan)
      FuncECPDF =  kmaxscan
   elseif (ECPDFType == SCANtextRED) then
      call getsrp (xMod, 'kmaxscan', kmaxscan)
      FuncECPDF =  kmaxscan
   elseif (ECPDFType == SCANtextGRN) then
      call getsrp (xMod, 'kmaxscanGRN', kmaxscan)
      FuncECPDF =  kmaxscan
   else
      call fatalerr(xMod, 'Unknown value for ECPDFType')
   end if
elseif (ECPDFChoice == 2) then
   call getsrtm (xMod, 'leafpresentwt', leafwtpresent, -99.)
   call getsrtm (xMod, 'stempresentwt', stemwtpresent, -99.)
   if (ECPDFType == FINTtext) then
      call getsrp (xMod, 'kminfint', kminfint)
      if (leafwtpresent < 0. ) then
         FuncECPDF = kminfint
      else
         call getsrp (xMod, 'betakminfint', betakminfint)
         call getsrp (xMod, 'kmaxfint',     kmaxfint)
         call getsrp (xMod, 'betakmaxfint', betakmaxfint)
         call getsrp (xMod, 'akfint', akfint)
         call getsrp (xMod, 'bkfint', bkfint)
         kminfint =  kminfint + betakminfint * (leafwtpresent/max(1.e-10, &
            leafwtpresent + stemwtpresent))
         kmaxfint =  kmaxfint + betakmaxfint * (leafwtpresent/max(1.e-10, &
            leafwtpresent + stemwtpresent))
         FuncECPDF = kminfint + (kmaxfint -kminfint)/(1.+exp(max(-30.,min(30.,&
            -akfint * (LeafArea - bkfint)))))
      end if
   elseif (ECPDFType == SCANtext) then
      call getsrp (xMod, 'kmaxscan', kmaxscan)
      if (leafwtpresent < 0. ) then
         FuncECPDF = kmaxscan
      else
         call getsrp (xMod, 'betakminscan', betakminscan)
         call getsrp (xMod, 'kminscan', kminscan)
```

```
            call getsrp (xMod, 'betakmaxscan', betakmaxscan)
            call getsrp (xMod, 'akscan', akscan)
            call getsrp (xMod, 'bkscan', bkscan)
            kminscan =  kminscan + betakminscan * (leafwtpresent/max(1.e-10, &
               leafwtpresent + stemwtpresent))
            kmaxscan =  kmaxscan + betakmaxscan * (leafwtpresent/max(1.e-10, &
               leafwtpresent + stemwtpresent))
            FuncECPDF = kminscan + (kmaxscan-kminscan)/(1.+exp(max(-30.,min(30., &
               -akscan * (LeafArea - bkscan)))))
         end if
      elseif (ECPDFType == SCANtextRED) then
         call getsrp (xMod, 'kmaxscan', kmaxscan)
         if (leafwtpresent < 0. ) then
            FuncECPDF = kmaxscan
         else
            call getsrp (xMod, 'betakminscan', betakminscan)
            call getsrp (xMod, 'kminscan', kminscan)
            call getsrp (xMod, 'betakmaxscan', betakmaxscan)
            call getsrp (xMod, 'akscan', akscan)
            call getsrp (xMod, 'bkscan', bkscan)
            kminscan =  kminscan + betakminscan * (leafwtpresent/max(1.e-10, &
               leafwtpresent + stemwtpresent))
            kmaxscan =  kmaxscan + betakmaxscan * (leafwtpresent/max(1.e-10, &
               leafwtpresent + stemwtpresent))
            FuncECPDF = kminscan + (kmaxscan-kminscan)/(1.+exp(max(-30.,min(30., &
               -akscan * (LeafArea - bkscan)))))
         end if
      elseif (ECPDFType == SCANtextGRN) then
         call getsrp (xMod, 'kmaxscanGRN', kmaxscan)
         if (leafwtpresent < 0. ) then
            FuncECPDF = kmaxscan
         else
            call getsrp (xMod, 'betakminscanGRN', betakminscan)
            call getsrp (xMod, 'kminscanGRN', kminscan)
            call getsrp (xMod, 'betakmaxscanGRN', betakmaxscan)
            call getsrp (xMod, 'akscanGRN', akscan)
            call getsrp (xMod, 'bkscanGRN', bkscan)
            kminscan =  kminscan + betakminscan * (leafwtpresent/max(1.e-10, &
               leafwtpresent + stemwtpresent))
            kmaxscan =  kmaxscan + betakmaxscan * (leafwtpresent/max(1.e-10, &
               leafwtpresent + stemwtpresent))
            FuncECPDF = kminscan + (kmaxscan-kminscan)/(1.+exp(max(-30.,min(30., &
               -akscan * (LeafArea - bkscan)))))
         end if
      else
         call fatalerr(xMod, 'Unknown value for ECPDFType')
      end if
   elseif (ECPDFChoice == 3) then
      call getsrp (xMod, 'ecpdf', ecpdf)
      if (ECPDFType == FINTtext) then
         FuncECPDF =  ecpdf
      elseif (ECPDFType == SCANtext) then
         call getsrp (xMod, 'alphaKscan', alphascan)
         call getsrp (xMod, 'betaKscan',  betascan)
         GroundCover = (100.*max(0., min(1., 1.-exp(max(-30., min(30.,&
            -ecpdf*LeafArea))))))
         if (GroundCover < 1.e-8) then
            GroundCover = 0.
         else
            GroundCover = max(0.,min(100.,alphascan * (GroundCover**betascan)))
         end if
         FuncECPDF =  (-log(max(1.e-3,1.-GroundCover/100.)))/max(1.e-10, LeafArea)
      elseif (ECPDFType == SCANtextRED) then
         call getsrp (xMod, 'alphaKscan', alphascan)
         call getsrp (xMod, 'betaKscan',  betascan)
         GroundCover = (100.*max(0., min(1., 1.-exp(max(-30., min(30.,&
            -ecpdf*LeafArea))))))
         if (GroundCover < 1.e-8) then
```

```
134            GroundCover = 0.
135         else
136            GroundCover = max(0.,min(100.,alphascan * (GroundCover**betascan)))
137         end if
138         FuncECPDF =  (-log(max(1.e-3,1.-GroundCover/100.)))/max(1.e-10, LeafArea)
139      elseif (ECPDFType == SCANtextGRN) then
140         call getsrp (xMod, 'alphaKscanGRN', alphascan)
141         call getsrp (xMod, 'betaKscanGRN',  betascan)
142         GroundCover = (100.*max(0., min(1., 1.-exp(max(-30., min(30., &
143            -ecpdf*LeafArea))))))
144         if (GroundCover < 1.e-8) then
145            GroundCover = 0.
146         else
147            GroundCover = max(0.,min(100.,alphascan * (GroundCover**betascan)))
148         end if
149         FuncECPDF =  (-log(max(1.e-3,1.-GroundCover/100.)))/max(1.e-10, LeafArea)
150      else
151         call fatalerr(xMod, 'Unknown value for ECPDFType')
152      end if
153   else
154      call fatalerr(xMod, 'Unknow value for ECPDFChoice')
155   end if
156   end function FuncECPDF
```