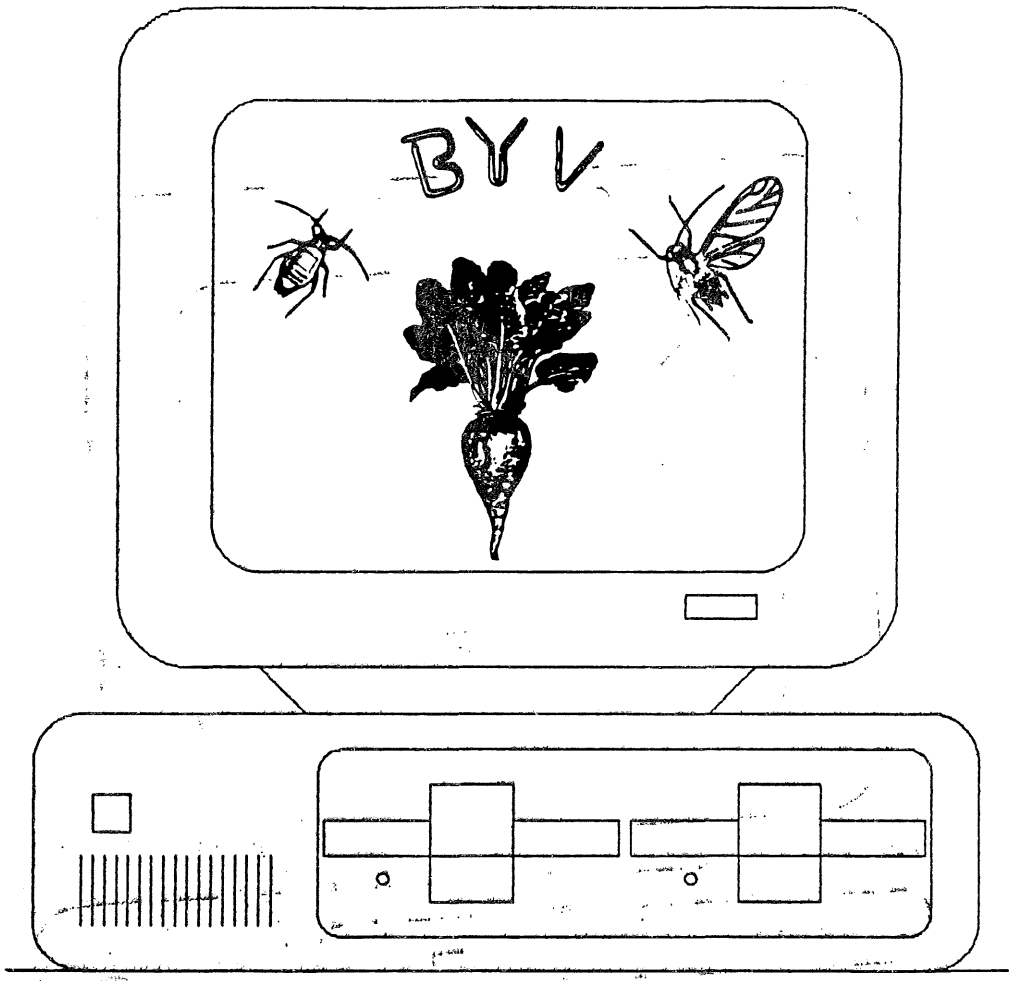
Een model voor de verspreiding van Beet Yellows Virus door Myzus persicae



Zesmaands doctoraalverslag van Wim Riesebos
in samenwerking met Corné Kempenaar
bij de vakgroep Theoretische Productie-ecologie
onder begeleiding van prof.dr.ir.R.Rabbinge en
ir.W.van der Werf

VOORWOORD

Dit verslag is het resultaat van een gemeenschappelijk uitgevoerd doctoraalvak door Corné Kempenaar en mij bij de vakgroep Theoretische Productie-ecologie. Corné heeft bij de vakgroep virologie een negenmaands doctoraalvak gedaan. Het onderwerp van onderzoek was het bietenvergelingsvirus (BYV). Corné heeft zes maanden onderzoek gedaan in suikerbietenveldjes. Dit om inzicht te verkrijgen in het verschil in verspreiding van BYV door bladluizen en in de vergeling van bieteplanten in velden met vroeg- en met laatgezaaide bieten. Corné stelde aan mij voor om samen een model te maken voor de verspreiding van BYV. De kennis die ik bij de vakgroep Theoretische Productie-ecologie en tijdens mijn negenmaands doctoraalvak informatica opgedaan had, zou hierbij goed van pas komen. Een beschrijving van het model en de simulatieresultaten zijn in dit verslag opgenomen. Dit verslag is het eindresultaat van mijn zesmaands doctoraalvak Theoretische Productie-ecologie. De theoretische achtergronden en de gebruikte veldproefresultaten worden in Corné zijn verslag "Systeemanalyse van de verspreiding van BYV in vroeg en in laat gezaaide suikerbieten" beschreven. Zijn inleiding is in dit verslag gedeeltelijk letterlijk overgenomen. Het betreft de hoofdstukken 1.1 en 1.2.

Het grootste deel van de achterliggende kennis die verwerkt is in het model is afkomstig uit het onderzoek van Corné Kempenaar en van Wopke van der Werf. Beide wil ik bedanken voor hun bijdragen aan het model. Rudy Rabbinge, die veel opbouwende kritiek heeft gegeven op fundamentele punten, wil ik hierbij bedanken voor zijn begeleiding.

Wim J.Riesebos

total *aphid-days* are on average 11500 and 5000 respectively. **Hence, the percentage infective *aphid-days* in late sown beet comes to an average of 48% and in early sown beet to an average of 18%.** The causes of the high percentage infective *aphid-days* in late sown beet are:

-a- a shorter latency period,

-b- a faster increase of the percentage infective leaf area.

(2) The *efficiëncy* depends on the number of infective *aphid-days*, of the susceptibility of the plants and of the rate at which the aphids disperse. **When the number of infective *aphid-days* (x) increases, the *efficiëncy* (E) decreases due to multiple infections ($E=4.37x^{-0.43}$).** The correlation between the number of infective *aphid-days* and the *efficiëncy* is the same in early and late sown beet. Causes of this similar correlation are:

-a- the almost similar way in which the aphids behave,

-b- the equal rate at which the aphids move,

-c- the equal susceptibility of the plants,

-d- the equal rate at which the aphids are predated.

(3) The decrease of the *efficiëncy* causes the number of yellowed plants (N) also to decrease ($N=4.37x^{0.57}$) with increasing numbers of infective *aphid-days*. **Due to the high numbers of infective *aphid-days* in late sown beet the number of yellowed plants is high ($N_{early} = 4.37 * 900^{0.57}$ ==> 211, $N_{late} = 4.37 * 5500^{0.57}$ ==> 592).** Numbers of yellowed plants counted in the field are of the same magnitude.

Samenvatting

De groene perzikbladluis (Myzus persicae) is de belangrijkste vector van het bietevergelingsvirus (BYV) in het veld. In laatgezaaide bieten zijn de aantallen bladluizen hoger dan in vroeggezaaide bieten. Hiermee samenhangend is de verspreiding van BYV groter. Met een computermodel is getracht de virusverspreiding en de vergeling van de planten te simuleren. Het doel was meer inzicht te verkrijgen in de oorzaken van de sterkere virusverspreiding in laatgezaaide bieten. Met het model zijn de volgende verbanden gevonden:

(1) het percentage infectieuze *bladluisdagen* hangt samen met de zaaidatum,

(2) de *efficiëntie* (a) hangt samen met (b) het aantal infectieuze *bladluisdagen*,

(3) het aantal vergeelde planten hangt samen met de zaaidatum.

(a) De efficiëntie is gedefiniëerd als het aantal vergeelde planten per infectieuze bladluisdag.

(b) Het aantal infectieuze bladluisdagen is gedefinieerd als de integraal van het aantal infectieuze bladluizen over de periode vanaf het uitzetten van de bladluizen tot het moment waarop inoculatie van een plant niet meer leidt tot de ontwikkeling van symptomen voor de oogst.

Het aantal vergeelde planten is het product van de efficiëntie en het aantal infectieuze bladluisdagen.

(1) Het aantal infectieuze *bladluisdagen* is in laatgezaaide bieten gemiddeld 5500 en in vroeggezaaide bieten gemiddeld 900. Het totale aantal *bladluisdagen* is gemiddeld respec-

tievelijk 11500 en 5000. **Het percentage infectieuze *bladluisdagen* is in laatgezaaide bieten dus gemiddeld 48% en in vroeggezaaide bieten gemiddeld 18%.** De oorzaken van het hogere percentage infectieuze *bladluisdagen* in laatgezaaide bieten zijn:

-a- een kortere latentieperiode,

-b- een snellere toename van het percentage besmet bladoppervlak.

(2) De hoogte van de *efficiëntie* is afhankelijk van het aantal infectieuze *bladluisdagen*, van de vatbaarheid van de planten en van de verspreidingssnelheid van de bladluizen. **Als het aantal infectieuze *bladluisdagen* (x) stijgt, dan neemt de *efficiëntie* (E) af door multipele infecties ($E=4.37x^{-0.43}$).** Het verband tussen het aantal infectieuze *bladluisdagen* en de *efficiëntie* is gelijk in laat- en in vroeggezaaide bieten. Oorzaken van dit gelijke verband zijn onder meer:

-a- het vrijwel gelijke loopgedrag van de bladluizen,

-b- de gelijke loopsnelheid van de bladluizen,

-c- de gelijke vatbaarheid van de bieteplanten,

-d- de gelijke predatiedruk.

(3) Door de afname van de *efficiëntie* vermindert ook de toename van het aantal vergeelde planten (N) bij stijgende aantallen infectieuze *bladluisdagen* ($N=4.37x^{0.57}$). **Door het hogere aantal infectieuze *bladluisdagen* in laatgezaaide bieten is het aantal vergeelde planten hoger ($N_{vroeg} = 4.37 * 900^{0.57}$ ==> 211, $N_{laat} = 4.37 * 5500^{0.57}$ ==> 592).** In veldproeven werden overeenkomstige aantallen vergeelde planten gevonden.

Hoofdstuk 1. Inleiding

1.1 Algemeen

In Nederland worden 2 virussen gevonden die vergelingsziekte veroorzaken in suikerbieten (Beta vulgaris). Dit zijn het bietevergelingsvirus of Beet Yellows Virus (BYV) en het zwakke bietevergelingsvirus of Beet Mild Yellowing Virus (BMYV). Buiten Nederland wordt nog een derde vergelingsvirus gevonden, het Beet Western Yellowing Virus (BWYV). BYV behoort tot de closterovirussen en BMYV en BWYV tot de luteovirussen. De virussen worden beschreven in een overzichtsartikel van Duffus (1973) en in het tekstboek over plantevirussen van Smith (1972).

In Nederland wordt ongeveer 134.000 ha suikerbieten geteeld. In de laatste jaren heeft de suikerbietenteelt in Nederland weinig last gehad van de vergelingsziekte, maar in bepaalde jaren was de veroorzaakte schade aanzienlijk (Heijbroek, 1984). Een reden waarom men in Nederland weinig last van vergelingsziekten heeft, is dat een gedeelte van het groeiseizoen van de suikerbieten in een periode valt waarin de infectiekansen klein zijn. Dit is het voorjaar. Als de planten vroeg in het seizoen (begin april) gezaaid kunnen worden, dan hebben de planten een vrij lange groeiperiode met een lage infectiekans. Bij eventuele infecties later in het seizoen valt de schade mee omdat de planten dan al vrij groot zijn. De laatste jaren worden proeven gedaan met het uitplanten van suikerbieten. Deze teelt geeft de planten een groeivoorsprong in vergelijking

met zaaien, waardoor de kans op schade door de ziekte kleiner wordt.

Voor de beschrijving van de symptomen en de verdeling van virus door de plant is de bouw van de plant belangrijk. De tweejarige suikerbiet staat één seizoen van ongeveer april tot oktober in het veld. In deze periode verdikt de hoofdwortel zich tot een opslagorgaan van suikers. Op de wortel staat een bladrozet die bestaat uit (bijna) volgroeide suikers-exporterende bladeren en uit het midden ontspringende suikers-importerende blaadjes. De assimilatenstroom gaat van de suikers-exporterende bladeren naar de wortel en naar de jonge, net ontspringende blaadjes. Op een gegeven moment zullen deze blaadjes ook suikers gaan exporteren. Per week kunnen er 2 tot 3 jonge bladeren bijkomen. Tijdens het groeiseizoen kunnen wel 50 bladeren gevormd worden. Beschrijvingen van de ontwikkeling en groei van suikerbieten staan in artikelen van Clark & Loomis (1978) en Milford et al. (1985).

Beide virussen veroorzaken vergelingssymptomen. BYV veroorzaakt op systemisch geïnfecteerde bladeren in eerste instantie nerfvergeling. Dit wordt vrij snel gevolgd door bladvergeling. De bladeren worden ook dikker en verliezen hun elasticiteit. Typisch voor BYV is het ontstaan van rode of bruine puntjes op het vergeelde blad. De symptomen van BMYV tonen zich minder agressief. De vergeling op de systemische bladeren begint tussen de nerven. De vergeling is eerst vrij licht en wordt vervolgens oranje/geel van kleur. Op zwaar vergeelde bladeren kunnen secundair pathogene schimmels gevonden worden. De symptomen van BWYV lijken

op die van BMYV. Naarmate de planten ouder worden, neemt de incubatieperiode toe. Voor infecties van jonge planten met BYV duurt het minimaal 10 dagen voor de symptomen zichtbaar worden. Aan het einde van het groeiseizoen hebben de planten een incubatieperiode van meer dan 8 weken. De incubatieperiode van BMYV is 1 tot 2 weken langer dan die van BYV. Aan de hand van de positie van de bladeren met symptomen en de corresponderende incubatieperiode kan de infectiedatum geschat worden (Van der Werf, 1988).

Op geïnoculeerde bladeren verschijnen gele vlekken op de plaats van infectie (de primaire infectieplaatsen). Na het blad binnen gekomen te zijn verspreiden de virussen zich met de assimilatenstroom door de plant. Op deze wijze komen de virussen in de wortel en in de jonge blaadjes terecht. De virussen komen hoofdzakelijk in het floëem voor. Over de beschadiging aan de wortel is weinig bekend. Over de systemische beschadiging aan de bovengrondse delen wel. Deze bestaat vooral uit een vermindering van het effectief assimilerend bladoppervlak. De schade hangt af van het moment van infectie (Heathcote, 1970; Watson et al., 1975).

De vergelingsvirussen komen ook op andere planten voor. BYV komt in Nederland naast de suikerbiet voor op kroten, spinazie en enkele onkruiden zoals vogelmuur (Russel, 1965). BMYV komt op minder planten voor, enkele waardplanten zijn weer kroten en enkele onkruiden, zoals herderstasje, hooornbloem, klein kruiskruid en vogelmuur (Duffus, 1973).

1.2 De epidemiologie van vergelingsvirussen

De verspreiding van de virussen vindt hoofdzakelijk plaats door bladluizen. De meest efficiënte factor voor beide virussen is Myzus persicae. Deze bladluis is polyfaag. Verder kan Aphis fabae BYV overbrengen. In de literatuur staan nog 20 bladluissoorten beschreven die vergelingsvirus kunnen overbrengen, maar de overdrachtsefficiëntie van deze bladluizen is zeer gering (Duffus, 1973).

BYV wordt op semi-persistente wijze overgebracht. Het virus wordt met het floëemsap opgezogen en adsorbeert aan de stylet van de bladluis. Bij opvolgende boringen van de bladluis kan het virus weer afgegeven worden. De bladluis blijft enkele dagen infectieus totdat er geen virus meer aan de stylet zit door afgifte of door vervelling. BMYV wordt op persistente wijze overgebracht. Het virus wordt met het floëemsap opgenomen en in de maag van de bladluis gekomen dringt het door de maagwand heen. Via het lichaamsvocht van de bladluis komt het virus in de speekselklieren. Van hieruit kan het virus bij opvolgende boringen weer afgegeven worden. De bladluis blijft infectieus, zelfs na vervelling, omdat het virus zich in de bladluis vermeerdert.

Voor de verspreiding is de biologie van M. persicae van belang (Hille Ris Lambers, 1972). Voor suikerbieten is van belang dat de cyclus in het veld begint met gevleugelde vrouwtjes die in het veld neerstrijken. Deze adulten zetten nymfen (alleen vrouwtjes) af, die in het L3 stadium vleugel-aanleg kunnen krijgen. Na vier vervellingen wordt het adulte stadium bereikt. Vooral de ongevleugelden zullen voor de opbouw van de populatie zorgen en voor de mogelijke

verspreiding van BYV in het veld. In de herfst worden gevleugelde vrouwtjes en mannetjes afgezet en bevrucht. Uit deze eieren komen in het voorjaar gevleugelde vrouwtjes die de zomerwaarden op gaan zoeken. Het aantal gevleugelden in het voorjaar hangt sterk af van de strengheid van de winter (Watson et al., 1975).

Met de bladluizen moeten de virussen in de nieuwe suikerbietenpercelen komen. De virussen worden niet met het zaad overgebracht. Zoals reeds eerder beschreven hebben beide virussen meerdere waarden. Uit deze waarden worden de virussen door bladluizen opgenomen. Vervolgens worden ze overgebracht naar de suikerbietenpercelen. Deze waarden zijn vooral de (wintervaste) onkruiden. Tot voor 15 jaar waren bietenkuilen grote bronnen van vergeling. Als in het voorjaar de bieten in deze kuilen uit begonnen te lopen konden bladluizen de virussen hieruit opnemen en overbrengen naar de percelen. Met het verdwijnen van de voederbietenkuilen zijn ook de vergelingsepidemiën verdwenen.

De cruciale momenten voor de verspreiding van de virussen zijn dus de winter en het voorjaar. De strengheid van de winter bepaalt het aantal overlevende bladluizen. Daarnaast hebben de virussen winterwaarden nodig. De mogelijkheden tot overwinteren van de virussen kunnen beperkt worden door bedrijfshygiëne, nl. het beperken van besmettingsbronnen zoals suikerbietenopslag en onkruiden. Op deze wijze kunnen de virussen slechts beperkt overgebracht worden naar de nieuwe percelen. Alleen als vroeg in het voorjaar bladluizen en virus in de percelen gesignaleerd worden, kan een

bespuiting met een bladluisdodend middel economisch verantwoord zijn.

1.3 Doel van het model

In vroeg ingezaaide bietenvelden komt minder vergeling voor door BYV dan in laat ingezaaide bietenvelden (zaaidatum-effect). Blijkbaar heeft de zaaidatum effect op enkele factoren die de verspreidingssnelheid van het virus beïnvloeden. De volgende factoren kunnen invloed hebben op de snelheid van virusverspreiding:

- latentieperiode (Kempenaar, 1987);
- incubatieperiode (Van der Werf, 1988);
- vatbaarheid van de bietenplant (Kempenaar, 1987);
- loopgedrag van M.persicae (Ferrar, 1968 en Jepson, 1983);
- maximum loopsnelheid van M.persicae (Ferrar, 1968);
- voorkeur van M.persicae voor jonge bladeren (Jepson, 1983);
- voorkeur van M.persicae voor vergeelde bladeren (Baker, 1960);
- retentieperiode van het virus in M.persicae (Sylvester, 1961);
- aantal M.persicae in het veld;
- predatiedruk op M.persicae;
- leeftijd van de planten;
- grootte van de planten.

Het is niet bekend hoe groot de invloed van deze factoren is op de snelheid waarmee het virus in het veld verspreid

wordt. Met als doel hier meer inzicht in te verkrijgen, is het model ontwikkeld.

Hoofdstuk 2. Het te modelleren biologische systeem

2.1 Algemeen

Het te modelleren biologische systeem omvat een bietenveldje van beperkte omvang (12 * 12m) en een populatie bladluizen (M. persicae) overeenkomstig de opzet van de veldproeven van Corné Kempenaar. De resultaten van deze veldproeven worden gebruikt voor validatie en callibratie van het model. De bladluizen zijn de enige belangrijke verspreiders van BYV.

De bladluizen:
Bladluizen vervellen 4 keer vanaf hun geboorte met tussenliggende periodes van ongeveer anderhalve dag. Na zes dagen zijn de bladluizen adult. Een infectieuze bladluis verliest haar infectiositeit met een vervelling. Een infectieuze bladluis die niet opnieuw het virus opneemt, blijft gedurende een periode van gemiddeld 12 uur (retentieperiode) infectieus. De bladluis moet minstens 30 minuten (Heathcote, 1964) achtereen zuigen op een besmet blad om het virus te verkrijgen. Om het virus op een plant over te dragen is waarschijnlijk een kortere zuigduur toerijkend (zie H3.3 de overdraagperiode). Een infectieuze bladluis die lang genoeg zuigt op een met virus besmet blad, is opnieuw gedurende een periode van 12 uur infectieus.

De bieteplanten:

Het opnieuw infecteren van een reeds geïnfecteerde plant sorteert geen enkel effect. Er ligt een periode tussen infectie en het opneembaar worden van het virus uit een plant (de latentieperiode). Geïnfecteerde planten vergelen na verloop van tijd. De periode tussen infectie en vergeling van de bieteplant (incubatieperiode) neemt toe gedurende het groeiseizoen. Alleen de bladeren die na infectie aan een plant verschijnen, bevatten het virus en worden infectieus na verloop van de latentieperiode. Planten die in een vroeg stadium geïnfecteerd worden, krijgen daardoor een groter percentage besmet bladoppervlak dan planten die in een laat stadium geïnfecteerd worden.

2.2 De grenzen van het systeem

Het bietenveldje met de bladluizenpopulatie wordt gezien als een afgerond geïsoleerd geheel. Er zijn geen interacties met de buitenwereld. Derhalve is er geen immigratie van bladluizen. Alleen gevleugelde bladluizen kunnen het systeem verlaten. De enige externe factor die van invloed is op de processen in het systeem is de temperatuur. De afmetingen van het bietenveldje vormen de grenzen van het systeem. De enige bladluizen in het veld zijn de bladluizen die uitgezet zijn en hun nakomelingen. De enige wijze waarop BYV in het veld kan komen, is via het uitzetten van infectieuze bladluizen.

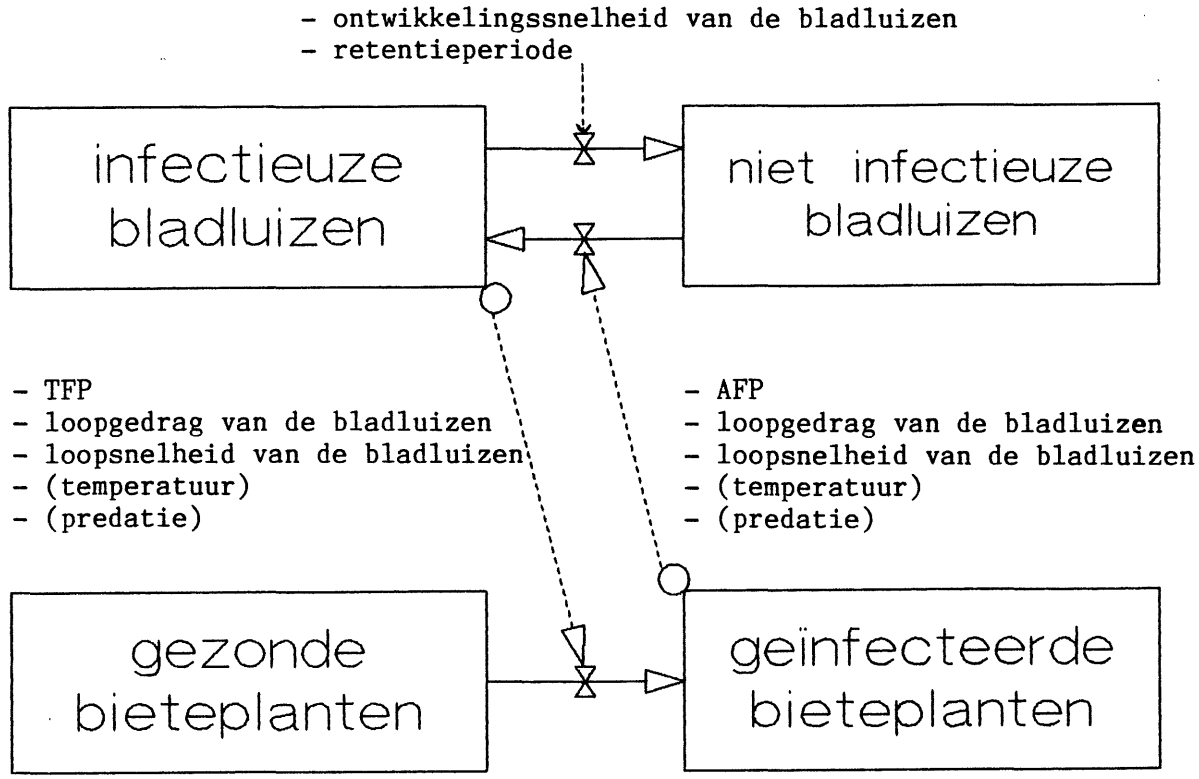
2.3 Het relatiediagram

Aan de hand van de bovenstaande systeembeschrijving, is het relatiediagram in figuur 1 opgesteld. Het relatiediagram is in drie delen opgesplitst. Het eerste deel (figuur 1a) geeft de hoofdinformatiestromen weer, het tweede deel (figuur 1b) de overgangen tussen de verschillende bladluisstadia en het derde deel (figuur 1c) de overgangen van de verschillende klassen van de bieteplanten.

De hoofdinformatiestromen:
De snelheid waarmee gezonde bieteplanten overgaan in besmette bieteplanten wordt bepaald door het aantal infectieuze bladluizen in de simulatie, de verspreidingssnelheid van de bladluizen, het loopgedrag, de transmissieperiode van het virus (TFP, Transmission Feeding Period), de predatiedruk en de temperatuur. De overgangssnelheid van gezonde bladluizen naar infectieuze bladluizen wordt bepaald door het totale besmette bladoppervlak van de bieteplanten, de verspreidingssnelheid van de bladluizen, het loopgedrag, de opneembaarheid van het virus (AFP, Aquisition Feeding Period), de predatiedruk en de temperatuur.

De overgangen tussen de verschillende stadia van de bladluizen:
De overgang van infectieus naar niet-infectieus geschiedt voor elke individuele bladluis na een retentie periode van 12 uur of na een vervelling. De overgangssnelheden tussen de verschillende bladluisstadia zijn onafhankelijk van de temperatuur verondersteld. De overgang van niet-infectieus naar infectieus is het resultaat van het zuigen van de

Figuur 1a

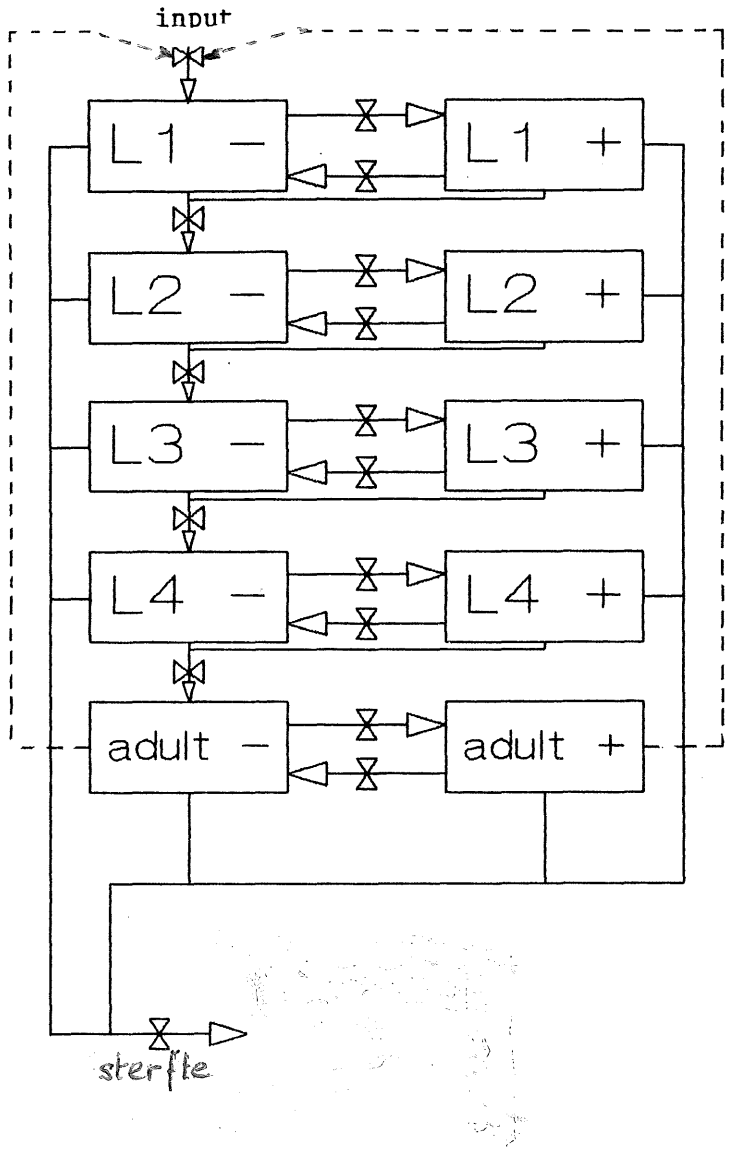


Figuur 1a. De verschillende factoren die invloed hebben op de mate van vergeling in het veld worden onderscheiden in factoren die een invloed hebben op de efficiëntie waarmee bladluizen het virus overdragen en in factoren die een invloed hebben op het aantal besmette bladluizen.

De mate van efficiëntie van virusoverdracht beïnvloedt de overgangssnelheid van gezonde planten naar geïnfecteerde planten.

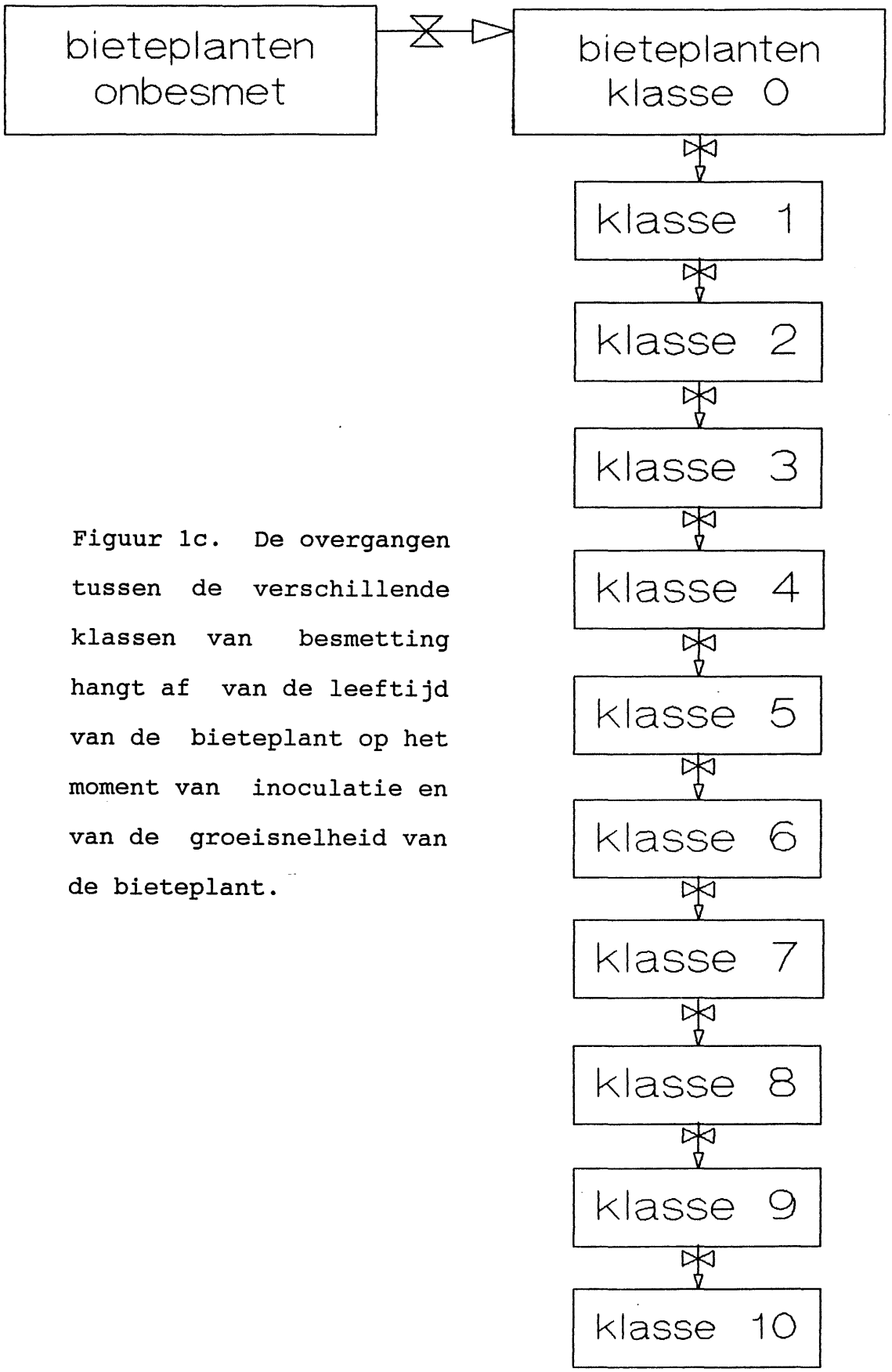
De relatieve snelheid waarmee infectieuze bladluizen overgaan in niet-infectieuze bladluizen is in het model constant. Factoren die invloed hebben op het aantal besmette bladluizen in de simulaties beïnvloeden dus de snelheid waarmee niet-infectieuze bladluizen infectieus worden.

Figuur 1b



Figuur 1b. De kans om gepredeerd te worden en daarmee uit de simulatie te verdwijnen is voor alle bladluizen even groot. De reproductiesnelheid is voor infectieuze en niet-infectieuze bladluizen even hoog. De overgangssnelheden tussen de infectieuze en niet-infectieuze bladluizen zijn voor elk bladluisstadium gelijk. Besmette bladluizen verliezen hun infectiositeit met een vervelling. Er komen nieuwe bladluizen in de simulatie bij indien het aantal bladluizen in de simulatie verhoogd moet worden.

Figuur 1c



Figuur 1c. De overgangen tussen de verschillende klassen van besmetting hangt af van de leeftijd van de bieteplant op het moment van inoculatie en van de groeisnelheid van de bieteplant.

bladluis op een besmette plant gedurende een tijd die minstens lang genoeg is om het virus uit de plant op te nemen (AFP). De sterftesnelheid van de bladluizen is constant in het model. Nakomelingen van infectieuze ouders zijn niet infectieus. Het virus beïnvloedt de reproductie-snelheid en de ontwikkelingssnelheid van de bladluizen niet.

De overgangen van de klassen van de planten:
De overgangssnelheden tussen de verschillende klassen waar de bieteplanten zich in kunnen bevinden is afhankelijk van de groeisnelheid en van de leeftijd van de bieteplant op het moment van inoculatie. De klasse van de bieteplant geeft aan hoeveel procent van het bladoppervlak besmet is. Zolang de latentieperiode niet voorbij is, kan het virus niet uit de plant opgenomen worden (ongeacht de klasse waar de plant zich in bevindt). Een bieteplant raakt besmet met het virus indien een infectieuze bladluis er lang genoeg op zuigt (TFP) om het virus over te dragen.

De temperatuur:
De temperatuur is van invloed op de groeisnelheid van de bieteplanten en de loopsnelheid van de bladluizen.

De predatiedruk:
De predatiedruk is van invloed op de leeftijdsverdeling van de bladluizen. De leeftijd van een bladluis bepaald samen met de temperatuur de snelheid waarmee een bladluis loopt. De verspreidingssnelheid van de bladluizen in het veld hangt daarom samen met de leeftijdsverdeling.

Hoofdstuk 3.

Het model

3.1 Modelkeuze

Het model bestaat uit twee gedeeltes. Het ene deel is deterministisch en beschrijft de groei en vergeling van de bieteplanten. Het andere deel is stochastisch en simuleert aan de hand van het loopgedrag van de bladluizen de verspreiding van BYV in het veld. De populatiedynamica van de bladluizen in het model is deels beschrijvend deels verklarend: Het aantal bladluizen in het veld wordt berekend op grond van veldtellingen. De uitkomst van de berekening hangt af van de verspreidingssnelheid van de bladluizen in de simulatie. De verspreidingssnelheid van de bladluizen is het resultaat van hun loopsnelheid en hun loopgedrag.

Het model is discreet. Dit wil zeggen dat de bladluizen en de bieteplanten in het veld niet gezien worden als een getal dat hun hoeveelheid weergeeft maar als afzonderlijke individuen (objecten). In het model worden twee soorten objecten onderscheiden, de bieteplant en de bladluis. De toestand van elk afzonderlijk object wordt beschreven door een aantal toestandsvariabelen. Van elke bladluis afzonderlijk kunnen daardoor, aan de hand van haar toestand en de toestand van de bieteplant waar zij zich op bevindt, haar loopgedrag en loopsnelheid bepaald worden.

De planten zijn in een tweedimensionaal coördinatenstelsel geplaatst (het veld). De bladluizen kunnen het veld lopend

niet verlaten. Aan de rand van het veld worden ze tegengehouden.

3.2 Aannames voor het model

De aannames zijn geordend van belangrijk naar minder belangrijk. De volgende aannames en vereenvoudigingen zijn voor het model gemaakt:

- Bladluizen lopen in een willekeurige richting. Er zijn aanwijzingen dat het loopgedrag van bladluizen beïnvloed wordt door jonge en vergeelde bladeren (Jepson, 1983 en Baker, 1960).
- Infectie van een bieteplant met BYV heeft geen invloed op de bladgroeisnelheid en de bladverschijningssnelheid van de plant. Het virus heeft hier in werkelijkheid wel invloed op (Van der Werf, 1988).
- De dichtheid van de bieteplanten heeft geen invloed op de verspreidingssnelheid van de bladluizen. In werkelijkheid zullen de bieten eerder aansluiten indien ze dicht opeen staan. De bladluizen beginnen zich in het model en waarschijnlijk ook in werkelijkheid pas effectief te verspreiden indien de bieten aansluiten.
- De verspreidingssnelheid van de bladluizen is dichtheidsonafhankelijk. Aangenomen wordt, dat de aantallen bladluizen in het systeem nooit zo hoog worden dat dit een motivatie tot verspreiding kan zijn (er is geen sprake van crowding).

- Gevleugelde bladluizen vliegen uit het systeem indien de temperatuur ligt tussen 15 en 30 graden Celcius. In werkelijkheid kunnen wegvliegende bladluizen slechts enkele meters verder weer in hetzelfde veldje neerstrijken.
- Ongevleugelde bladluizen verlaten het systeem alleen door sterfte. In werkelijkheid kunnen de bladluizen het veld ook lopend verlaten. In de simulaties bereiken relatief weinig bladluizen de grenzen van het veld. Dat deze bladluizen het systeem niet kunnen verlaten, heeft daardoor waarschijnlijk weinig invloed op de simulatieresultaten.
- De bladluizen ondervinden geen enkele invloed van de aanwezigheid van het virus. Ze gaan bijv. niet sneller lopen of langzamer vervellen. Er zijn geen aanwijzingen dat dit wel zo zou zijn.
- De afstand tussen de rijen in het veld wordt constant verondersteld. De afstand van de bieteplanten binnen een rij is ook telkens even groot. Aangenomen wordt dus dat de bieteplanten homogeen verdeeld zijn over het veld.
- Dag en nacht hebben geen invloed op het doen en laten van de bladluizen, tenzij indirect door temperatuursschommelingen. De hoogte van de temperatuur is in het model van invloed op de loopsnelheid van de bladluizen maar niet op de loopfrequentie.
- De luchtvochtigheid heeft geen invloed op het gedrag van de bladluis. In het model zijn geen gegevens over de luchtvochtigheid opgenomen.

- Het geïnoculeerde blad wordt niet meegenomen in de berekening van het percentage besmet bladoppervlak van de plant. Deze vereenvoudiging zal een kleinere procentuele afwijking ten opzichte van het werkelijke percentage besmet bladoppervlak geven naarmate dit percentage hoger is.
- De bladluizen reageren momentaan op veranderingen in de temperatuur.
- De bieteplanten reageren momentaan op veranderingen in de temperatuur.
- Alle bieteplanten groeien even snel. In werkelijkheid zijn er verschillen in groei tussen de planten onderling.
- Alle bieteplanten zijn even oud. Ze zijn op dezelfde datum gezaaid.
- De groeicondities zijn voor de bieteplanten optimaal. Mineralen en water zijn ongelimiteerd aanwezig.
- Bij een vervelling raakt de bladluis het virus kwijt omdat het virus aan de stilet adsorbeert. Bij een vervelling vervelt ook de stilet (ectodermaal weefsel).
- Predatiedruk is constant. De werkelijke predatiedruk in het veld is niet gemeten. Bij een constante predatiedruk wordt de leeftijdsverdeling onder de bladluizen in het veld redelijk goed benaderd.
- Ontwikkelingssnelheid van de bladluis is temperatuursonafhankelijk. De ontwikkelingssnelheid is in werkelijkheid wel temperatuursafhankelijk. Verwacht wordt dat de ontwikkelingssnelheid slechts weinig invloed heeft op het percentage infectieuze bladluizen.

- Ontwikkelingssnelheid van de bladluis is onafhankelijk van de waardplantkwaliteit.
- Alle bladluizen hebben dezelfde kans om gepredeerd te worden. Predatoren hebben geen voorkeur voor bladluizen van een bepaald stadium.
- De bladluizen hebben een vaste maximum leeftijd. Bij de predatiedruk die in het model gehanteerd wordt, wordt de maximum leeftijd nooit bereikt. Deze aanname is pas van invloed op de simulatieresultaten indien de maximum leeftijd wel door veel bladluizen bereikt wordt.
- Er sterven geen bieteplanten. Sterfte onder de bieteplanten is in werkelijkheid zeer gering.
- Er is geen vraat aan de bieteplanten. In het veld is geen vraat van betekenis gevonden.
- Bladluissoort Myzus persicae is de enige vector van BYV. Andere bladluissoorten die BYV over kunnen brengen, doen dit met een efficiëntie die veel lager is dan die van Myzus persicae.
- Er zijn alleen vrouwelijke bladluizen in de populatie. Ook in werkelijkheid zullen er tijdens het groeiseizoen geen mannetjes gevonden worden. Mannelijke bladluizen verschijnen pas tegen de herfst.
- Bladluisaantallen in het veld worden berekend uit bladluizentellingen.
- De temperatuur in de simulatie wordt bepaald aan de hand van temperatuurstabellen over de maanden april t/m oktober '86 van het nabijgelegen meteostation Swifterbant (vakgroep Natuur- en Weerkunde).

3.3 Modelbeschrijving

In deze paragraaf wordt toegelicht hoe de verschillende factoren, die invloed hebben op de verspreidingssnelheid van BYV in het veld, in het model verwerkt zijn.

- De latentieperiode

De latentieperiode neemt gedurende het groeiseizoen toe van 4 tot 12 dagen. Voor het model wordt een lineair verband verondersteld tussen de ouderdom van de bieteplanten en de lengte van de latentieperiode. Bij de jongste bieteplanten (theoretisch nul dagen) is de lengte van de latentieperiode 4 dagen. Bij de oudste bieteplanten (in het model 7 maanden) 12 dagen.

- De acquisitieperiode

Vijf minuten is te kort om het virus uit de plant op te nemen maar na een periode van 30 tot 60 minuten kunnen bladluizen het virus uit de plant opgenomen hebben (Heathcote, 1964). In proeven wordt de acquisitieperiode bijna altijd als volgt bepaald. Bladluizen worden op een besmette plant gezet. Na een bepaalde tijd worden ze er af gehaald. Getest wordt vervolgens hoe groot het percentage bladluizen is, dat het virus bij zich draagt. Dit soort proeven zegt alleen wat over de periode dat de bladluizen minimaal moeten verblijven op een besmette plant om het virus op te nemen. De gemiddelde werkelijke zuigtijd kan hieruit niet bepaald worden. Gemiddelde, omdat in het model voor elke bladluis dezelfde acquisitietijd gehanteerd wordt.

Werkelijke zuigtijd, omdat in het model de periode ingaat bij het insteken van de zuigbuis in het blad en eindigt met het er weer uittrekken. Voor het model is een periode van 2 uur genomen.

- De transmissieperiode

Hierover zijn weinig concrete aanwijzingen gevonden in de literatuur. Volgens specialisten (W. van der Werf) is de periode om virus aan de plant af te kunnen geven kleiner dan de periode nodig om het virus uit de bieteplant op te nemen. Voor het afgeven van het virus voldoet in principe elk floëemvat. Om het virus op te nemen moet een floëemvat aangeboord worden met het virus. Indien niet alle floëemvaten het virus bevatten, kan een bladluis meerdere keren moeten boren voordat een floëemvat getroffen wordt met het virus. In het model moet de overdraagperiode in een geheel aantal uren gegeven worden, daarom is deze periode op één uur gesteld.

- Vervellen van de bladluis

De periode tussen twee vervellingen is voor elk larvale stadium bij benadering gelijk. De gemiddelde periode tussen twee vervellingen ligt tussen de anderhalve en twee dagen, bij een temperatuur van rond de 20 graden Celcius (Jansson and Smilowitz, 1985). Voor het model is een gemiddelde duur van anderhalve dag genomen tussen de vervellingen ongeacht de temperatuur.

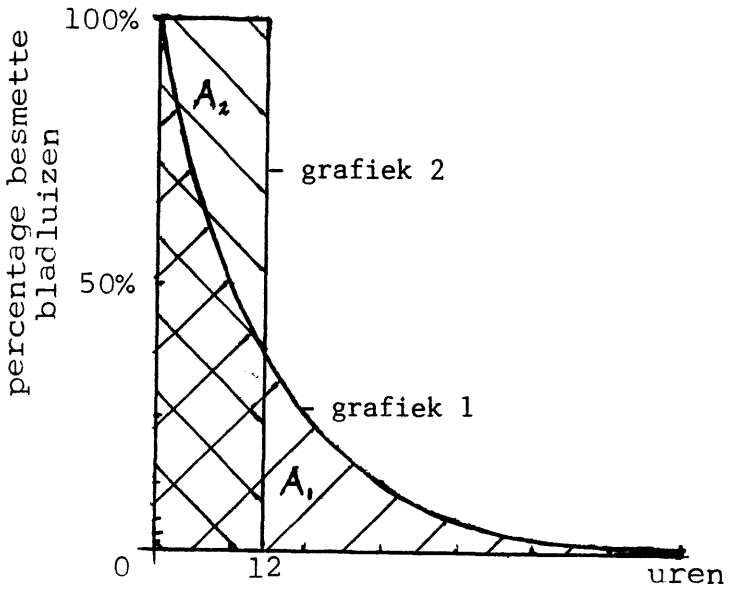
- Retentieperiode

De halfwaardetijd van BYV in Myzus persicae is 8.3 uur (Sylvester, 1961). Bij een halfwaardetijd van 8.3 uur is de gemiddelde tijd dat een bladluis het virus bij zich heeft 12 uur (zie figuur 2).

- Mortaliteit en predatiedruk

Bladluizen sterven door predatie en door het bereiken van de maximum leeftijd. In het model is de maximum leeftijd 42 dagen. Alleen bij zeer lage predatiedruk bereiken de bladluizen de maximum leeftijd. Bij een constante predatiedruk van 24% van het totaal aantal bladluizen per dag is de gemiddelde leeftijd van de bladluizen ongeveer 4 dagen. Omdat de leeftijdsverdeling van de bladluizen in de veldproeven bijna constant is (zie tabel 1a en 1b), is in het model voor een constante predatiedruk gekozen. Bij deze constante predatiedruk wordt de leeftijdsverdeling van de bladluizen in de veldproef van Corné Kempenaar benaderd.

Figuur 2



Figuur 2. De halfwaardetijd van BYV is 8.3 uur. Indien van een groep bladluizen die allemaal gelijktijdig met het virus besmet zijn het percentage infectieuze bladluizen uitgezet wordt tegen de tijd, krijgen we grafiek 1. Het oppervlak onder deze lijn (A_1) wordt uitgedrukt in besmette bladluisdagen. Dit is voor het model een belangrijke eenheid. De mate van vergeling in het veld hangt hier nauw mee samen. In het model hebben de bladluizen allemaal dezelfde retentieperiode. Alle bladluizen blijven gedurende de lengte van de retentieperiode infectieus. Na de retentieperiode worden alle bladluizen gelijktijdig niet-infectieus (grafiek 2). De retentieperiode moet zo lang zijn, dat het aantal besmette bladluisdagen onder grafiek 2 (A_2) gelijk is aan het aantal besmette bladluisdagen onder grafiek 1 (A_1, $A_1 = A2$).

$$A_1 = 8.3 \int_0^{\infty} 2^{-t}\, dt \quad ==> \quad 8.3[2^{-t}/\ln 2] \quad ==> \quad 8.3*1.44 = 12.0$$

$$A_2 = 1 * 12.0 = 12.0$$

Tabel 1a.

Leeftijdsverdeling van de bladluizen in het veld:

L4 staat voor L4 ongevleugeld.

In het vroeg gezaaide gewas (18 april 1986).

	5/7	11/7	17/7	24/7	30/7	gemiddeld
L1-L3	86.4	55.2	65.2	81.9	53.3	66.5 (79.5)
L4	3.2	8.7	8.1	4.5	13.3	6.1 (7.3)
adult	3.1	11.4	18.7	8.7	26.7	11.0 (13.2)

In het laat gezaaide gewas (20 mei 1986).

	5/7	11/7	17/7	24/7	30/7	gemiddeld
L1-L3	56.7	63.1	55.9	50.1	27.2	55.7 (68.0)
L4	12.5	8.8	6.8	12.9	17.1	10.2 (12.5)
adult	3.3	22.2	20.9	18.7	27.1	16.0 (19.5)

Het totaal van de percentages van de kolommen van de veldtellingen is niet gelijk aan 100 omdat de L4 gevleugeld, die Corné Kempenaar getelt heeft, weggelaten zijn. De omgewerkte getallen tellen wel op tot 100.

Tabel 1b.

Leeftijdsverdeling van de bladluizen in de simulatie:

L4 staat voor L4 ongevleugeld.

In het vroeg gezaaide gewas (18 april 1986).

	7/7	14/7		21/7	28/7	gemiddeld
L1-L3	53	79		62	57	63
L4	15	6		16	12	12
adult	32	15		22	31	25

In het laat gezaaide gewas (20 mei 1986).

	7/7	14/7		21/7	28/7	gemiddeld
L1-L3	63	76		68	54	65
L4	7	9		15	13	11
adult	29	15		17	32	23

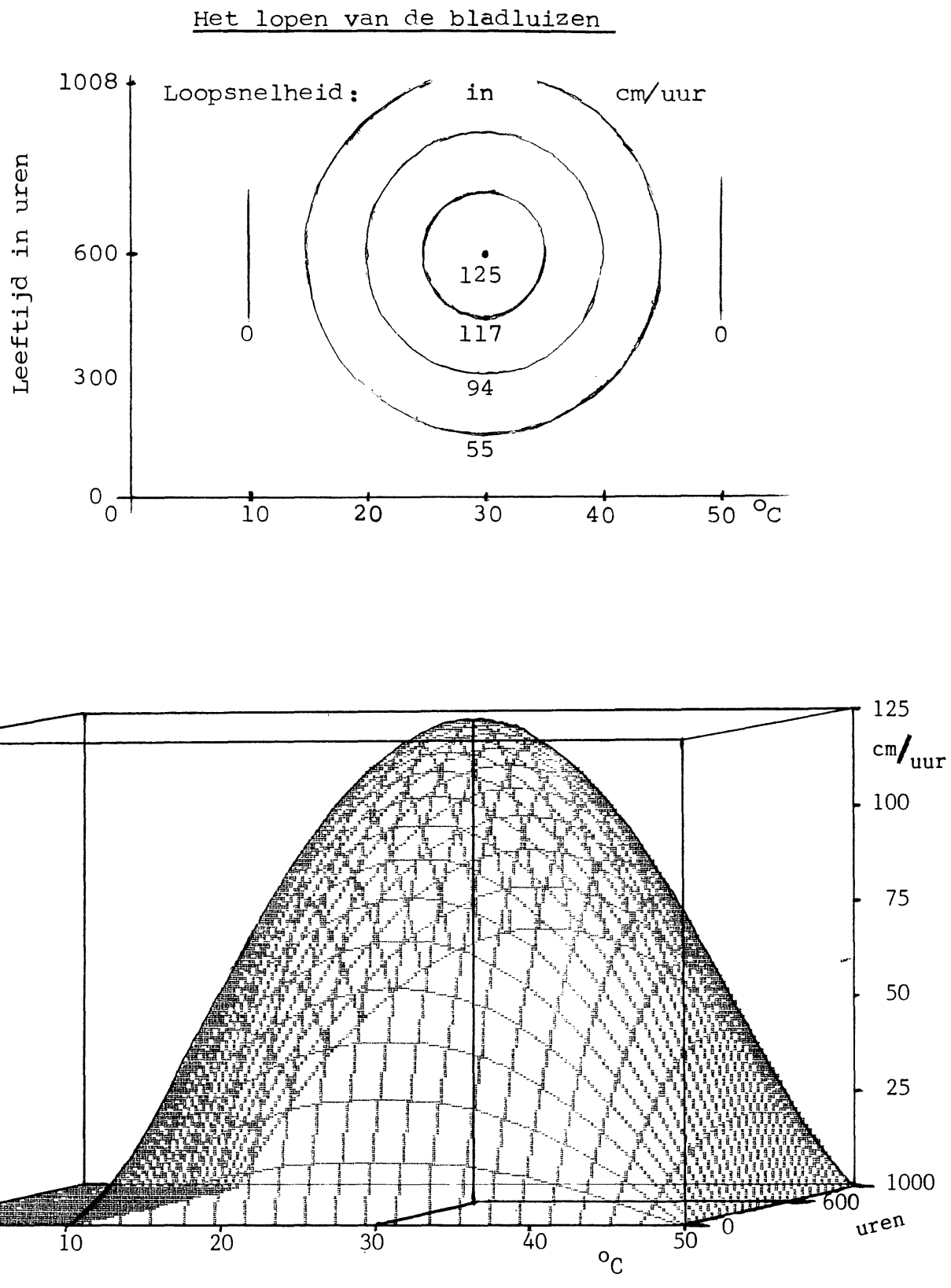
- Het lopen van de bladluizen

Over de snelheid waarmee bladluizen zich in het veld verplaatsen is weinig bekend. Er zijn wel proeven gedaan om meer over de loopsnelheid van de bladluizen te weten te komen. De bladluizen liepen daarbij over de grond (Ferrar, 1968). Voor het model is het belangrijk te weten hoe snel de bladluizen zich in een bietenveld kunnen verplaatsen via de bladeren en bladbruggen van de planten. Hierover is geen informatie gevonden.
In het model is de snelheid waarmee de bladluizen lopen alleen afhankelijk van de temperatuur en hun leeftijd. De looprichting is volslagen willekeurig.
De bladluizen lopen niet aan een stuk door. Ze verrichten proefboringen en ze moeten zuigen om voedsel op te nemen. Gedurende deze activiteiten zijn de bladluizen in rust en leggen dus geen afstand af. De verspreidingssnelheid van de bladluizen hangt af van de loopsnelheid, de loopfrequentie en de looprichting.

- De loopsnelheid

De minimum, optimum en maximum looptemperatuur voor de bladluis zijn respectievelijk 10, 30 en 50 graden Celcius. De leeftijd van de bladluis waarbij de maximum loopsnelheid bereikt wordt, is 25 dagen oftewel 600 uur. De maximum loopsnelheid bedraagd bij 30 graden Celcius 125 cm/uur.

Figuur 3



Figuur 3. De loopsnelheid van de bladluizen is afhankelijk van de omgevingstemperatuur en de leeftijd van de bladluizen. De maximum loopsnelheid van 125 cm/uur wordt bereikt bij een temperatuur van 30 graden Celcius en een bladluisleeftijd van 600 uur.

Grafiek 3 geeft het verband weer tussen de temperatuur, de leeftijd en de loopsnelheid van de bladluis. In formule:

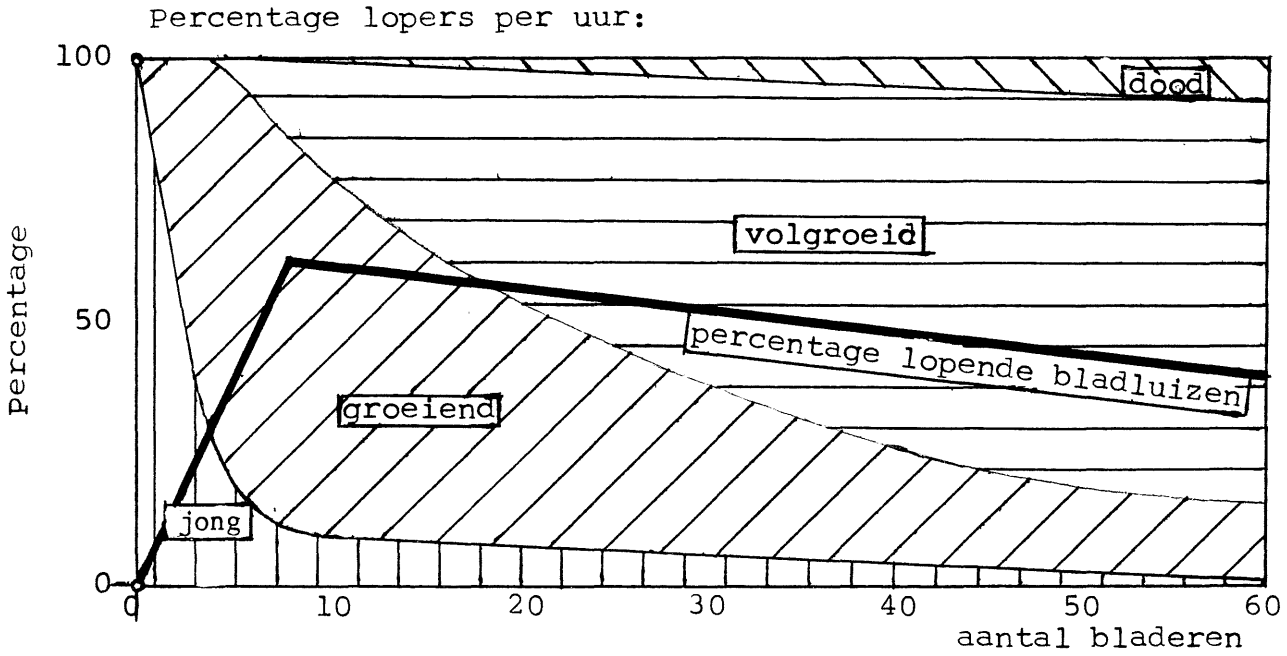
$$\rightarrow V = \left(125 - \frac{(age-600)^2}{2880}\right) * \left(1 - \frac{(temp-30)^2}{400}\right) \text{ cm/uur.}$$

- De loopfrequentie

De bladluizen zijn niet voortdurend in beweging. De motivatie om te lopen of juist niet is niet goed bekend. Vast staat, dat bladluizen, die op een waardplant zitten waar goed voedsel uit gehaald kan worden, weinig lopen en veel zuigen. Op een slechte waardplant zuigen de bladluizen minder en lopen meer. De bladluizen zoeken naar een goede plek door telkens proefboringen te verrichten. Een proefboring duurt waarschijnlijk (Van der Werf) niet lang genoeg om het virus uit de plant op te nemen of er aan af te geven. Uit niet gepubliceerde bevindingen van P.C.Jepson is afgeleid dat:

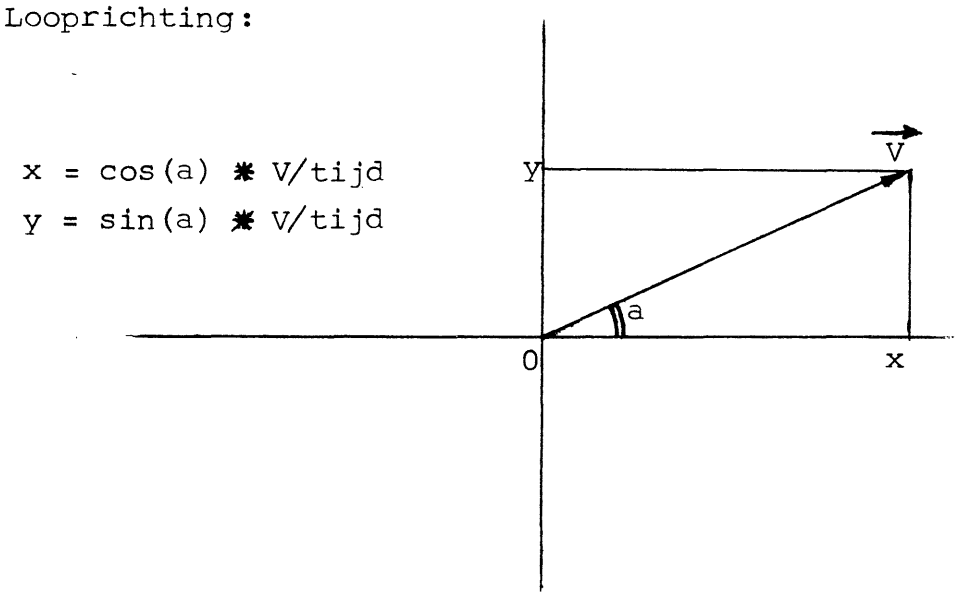
- de jongste hartblaadjes zeer geschikt zijn als voedselbron voor bladluizen en dat een bladluis gezeten op zo'n blad geen aanleiding heeft om te lopen. In het model lopen de bladluizen die zich op een jong blaadje bevinden niet.
- groeiend blad een lage voedingswaarde voor de bladluizen heeft. In het model loopt 67% van alle bladluizen, die op groeiend blad zitten, binnen een uur.
- volgroeid bietenblad een iets betere voedingswaarde heeft. In het model loopt 33% van alle bladluizen, die op volgroeid blad zitten, binnen een uur.

Figuur 4



Figuur 4. Het bladoppervlak van een bieteplant kan onderverdeeld worden in jong, groeiend, volgroeid en dood blad. In de figuur is het aandeel van de verschillende categoriën in procenten weergegeven als een functie van het aantal bladeren aan de plant. Het percentage bladluizen dat per uur loopt, hangt af van de verdeling van het bladoppervlak over de verschillende categoriën.

Figuur 5



Figuur 5. Uit de loopsnelheid, de tijdstap en de hoek waaronder de bladluis loopt kan de afgelegde afstand in de x en de y richting bepaald worden.

- dood bietenblad de minste voedingswaarde voor bladluizen heeft. Alle bladluizen die zich op dood blad bevinden lopen in het model binnen een uur.

Voor het hele groeiseizoen van de bieteplanten is bekeken hoeveel jong, groeiend, volgroeid en dood blad er aan de plant zit (figuur 4). Aan de hand hiervan en aan de hand van bovengenoemde aannames kan berekend worden hoeveel procent van het totaal aantal bladluizen op een bepaald tijdstip in het model per uur zal lopen (zie grafiek in figuur 4).
Vergeelde bladeren hebben doorgaans een goede voedingswaarde voor de bladluizen. Het effect van vergeelde planten in het veld op de loopfrequentie van de bladluizen is in het model buiten beschouwing gelaten.

In formule:

%lopers/uur = 0.33*%volgroeid blad +
0.67*%groeiend blad +
1.00*%dood blad.

Bovenstaande formule is alleen geldig indien de bladluizen random verdeeld zijn over de bieteplant.

- De looprichting

De richting waarin een bladluis loopt is willekeurig en wordt daarom elke tijdsstap opnieuw gekozen uit een uniforme verdeling van 0 tot 360 graden (α). De bladluizen hebben in het model geen voorkeursrichting en worden niet aangetrokken door vergeeld en jong bietenblad.

Voor elke bladluis wordt met behulp van een kanstrekking bepaald of ze loopt of zuigt. Indien de bladluis loopt, kan de afgelegde afstand in de x en de y-richting in het veld berekend worden uit de loopsnelheid, de lengte van de tijdsstap en de hoek (α) waaronder de bladluis loopt (figuur 5).

- het verband tussen de tijdstap en de gemiddelde lineaire verplaatsing van de bladluizen

Als maat voor de verspreiding van de bladluizen (S) wordt de gemiddelde lineaire afstand van de bladluizen tot de plaats van uitzetten genomen. Van invloed op deze gemiddelde afstand zijn:

-a- de grootte van de tijdstap (t),

-b- de duur van de simulatie (T),

-c- de loopsnelheid van de bladluizen (V),

-d- de fractie bladluizen die loopt (p).

Het aantal bladluizen is niet van invloed op de mate van verspreiding.

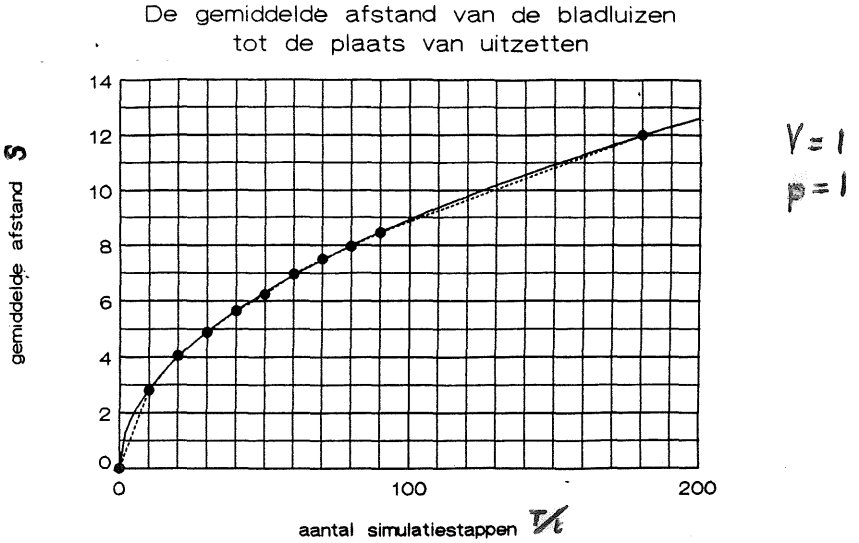
Het volgende empirische verband tussen de gemiddelde lineaire afstand van de bladluizen en bovengenoemde factoren is gevonden (Sabelis, 1981):

$S = 0.9 * V * t * (p * T/t)^{0.5}$.

Als de tijdstap een factor q groter of kleiner wordt gemaakt dan wordt de gemiddelde lineaire afstand (S_q):

$S_q = 0.9 * V * q*t * (p * T/(q*t))^{0.5}$ ==>
$S_q = q/q^{0.5} * 0.9 * V * t * (p * T/t)^{0.5}$.
$(q/q^{0.5} == q^{0.5})$

Hieruit blijkt dat V met een factor $q^{0.5}$ moet toenemen om dezelfde mate van verspreiding te behouden ($S = q^{0.5} * S_q$).



- Groei van de bieteplanten

Zie ook het eerste deel van de vierdelige artikelreeks "An analysis of leaf growth in sugar beet" (Milford,G.F.J., T.O.Pocock and J.Riley, 1985) . Het simuleren van de groei van de bieteplanten begint in het model pas dan als er 20 dagen verlopen zijn na het zaaien. De twee cotylen die eerst boven de grond verschijnen worden niet gesimuleerd. Deze twintig dagen worden gehanteerd onafhankelijk van de

omgevingstemperatuur. De bladverschijningssnelheid en de bladgroeisnelheid zijn rechtlijnig afhankelijk van de temperatuur. Niet alle bladeren kunnen in het model even groot worden. Rond het tiende blad ligt een optimum. Het verschil in afmeting wordt verondersteld een gevolg te zijn van een verschil in de ontwikkelingsduur en de groeisnelheid tussen de bladeren. Beide factoren leveren een even groot aandeel hierin. Elk blad heeft een eigen groeifactor (waarde tussen 0 en 1) welke de uiteindelijke grootte van het blad beïnvloedt. De optimum temperatuur voor bladgroei, bladverschijning en ontwikkelingsduur is in het model 24 graden Celcius.

- Bladverschijning (figuur 6)

Het aantal gevormde bladeren in bladeren per dag:
3.16e-2 * temperatuur - 3.16e-2

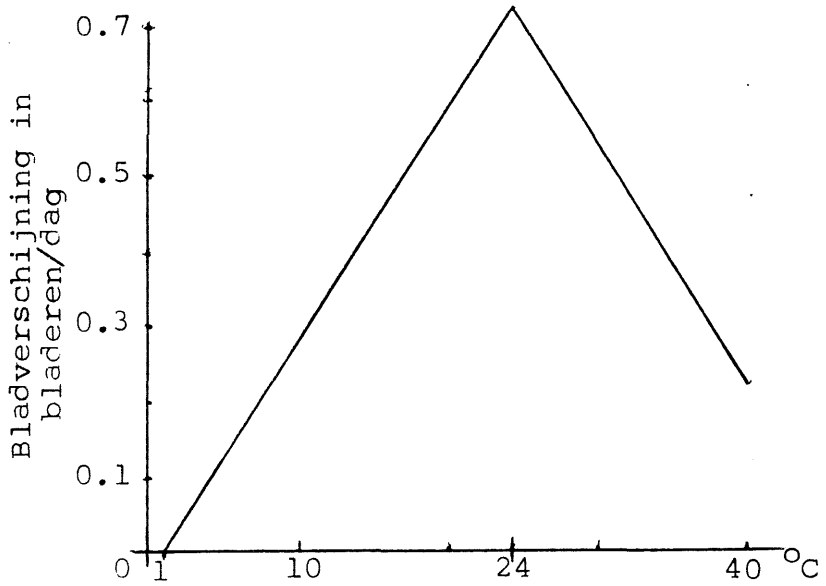
- Bladgroei (figuur 7)

Het gevormde bladoppervlak in cm^2 per dag:
groeifactor * (1.1 * temperatuur -3.3)

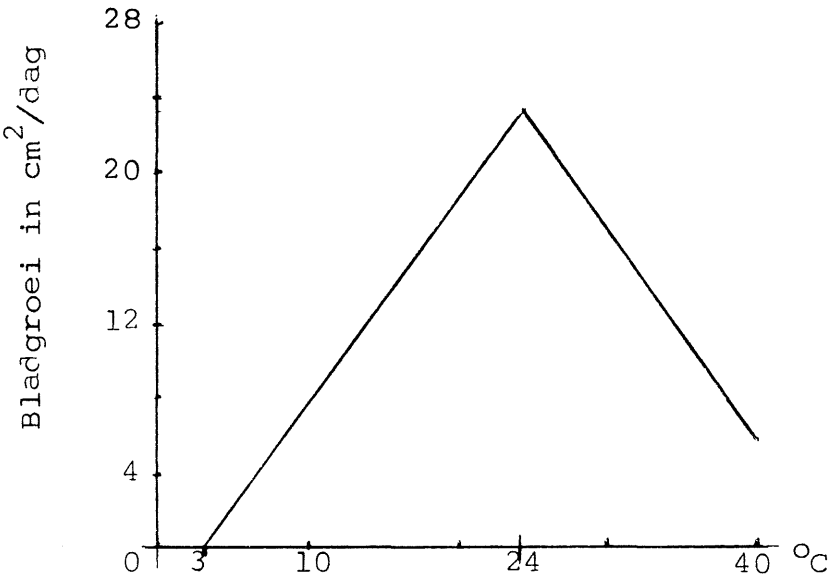
- Ontwikkelingsduur (figuur 8)

De ontwikkelingsduur van het bietenblad in graaddagen:
groeifactor * (4.25 * temperatuur + 355)

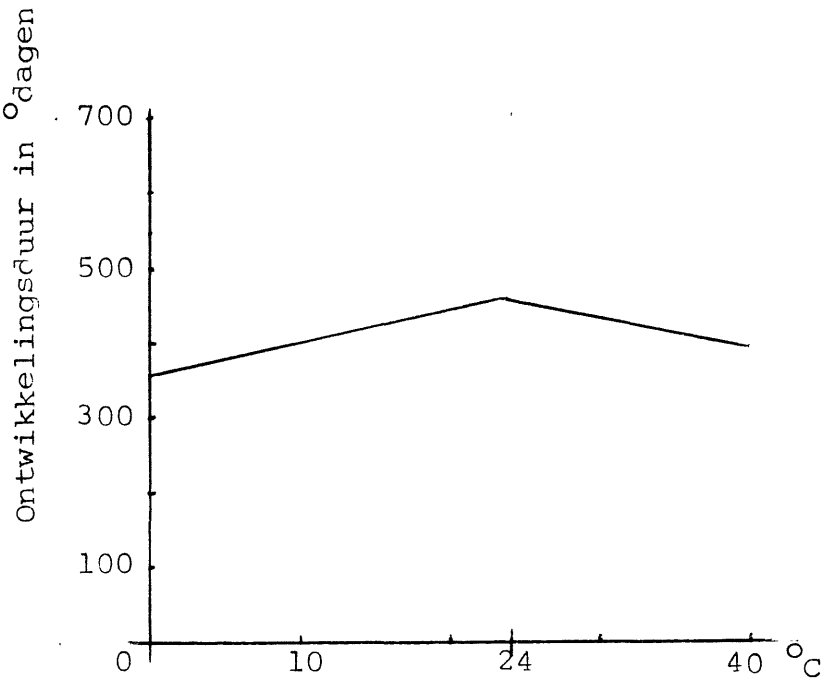
Figuur 6



Figuur 7



Figuur 8



- Berekenen van de aantallen bladluizen tijdens de simulatie

De bladluizentellingen van Corné Kempenaar, worden in het model gebruikt om het aantal bladluizen in de simulatie te berekenen. In het veld zijn op 9 planten bladluizen getelt. De getelde aantallen zijn omgewerkt naar een schatting voor het aantal bladluizen op de middelste 39 planten (Kempenaar, 1987). Het aantal bladluizen op de middelste 39 planten in de simulatie wordt met dit geschatte aantal vergeleken. Wijkt het aantal bladluizen in de simulatie meer dan 10 procent af van het geschatte aantal, dan wordt het aantal bladluizen in de simulatie aangepast. Op elke plant in de simulatie wordt het aantal bladluizen daartoe met een bepaalde factor vermenigvuldigd. Na deze aanpassing is het aantal bladluizen op de middelste 39 planten gelijk aan het geschatte aantal bladluizen.

Door deze manier van bladluisaantallen berekenen wordt het aantal bladluizen in de simulatie groter indien de verspreidingssnelheid van de bladluizen groter is: Bij een hoge verspreidingssnelheid van de bladluizen daalt het aantal bladluizen op de middelste 39 planten in de simulatie snel. Hierdoor zijn de schattingen voor het aantal bladluizen in het veld op deze middelste 39 planten veel hoger dan het in de simulatie getelde aantal. Het aantal bladluizen in de simulatie moet daarom sterk verhoogd worden om dit verschil te nivelleren. Door herhaalde sterke verhogingen van het aantal bladluizen in de simulatie kan het aantal bladluizen zeer hoog worden.

Hoofdstuk 4. Resultaten

4.1 Inleiding

Voor de latentieperiode, de voorkeur van bladluizen voor jonge bladeren, de efficiëntie van virusoverdracht, de maximum loopsnelheid van de bladluizen, de predatiedruk en het percentage lopers per uur is de invloed op de virusverspreidingssnelheid bepaald. Ook zijn er simulaties gedaan waarbij de bladluizentellingen van laatgezaaide bieten gebruikt zijn in combinatie met vroeggezaaide bieten en vice versa. Dit laatste om het effect dat het aantal bladluizen in de simulaties op het aantal vergeelde planten heeft, bij verschillende gewascondities te bepalen. Van alle bovengenoemde factoren is de invloed op het aantal infectieuze bladluisdagen en de invloed op de efficiëntie waarmee de bladluizen het virus verspreiden bepaald.

Infectieuze bladluisdagen:
Het aantal infectieuze bladluisdagen kan op twee manieren veranderd worden. Enerzijds door het totale aantal bladluizen in de simulatie te vergroten of te verkleinen (bij een zelfde percentage infectieuze bladluizen) en anderzijds door het verhogen of verlagen van het percentage infectieuze bladluizen (bij een zelfde totaal aantal bladluizen). Meestal heeft een factor (zie H1.3) invloed op zowel het aantal bladluizen als het percentage infectieuze bladluizen. Het effect dat een factor heeft op het aantal

infectieuze bladluisdagen wordt in aparte grafieken uitgezet.

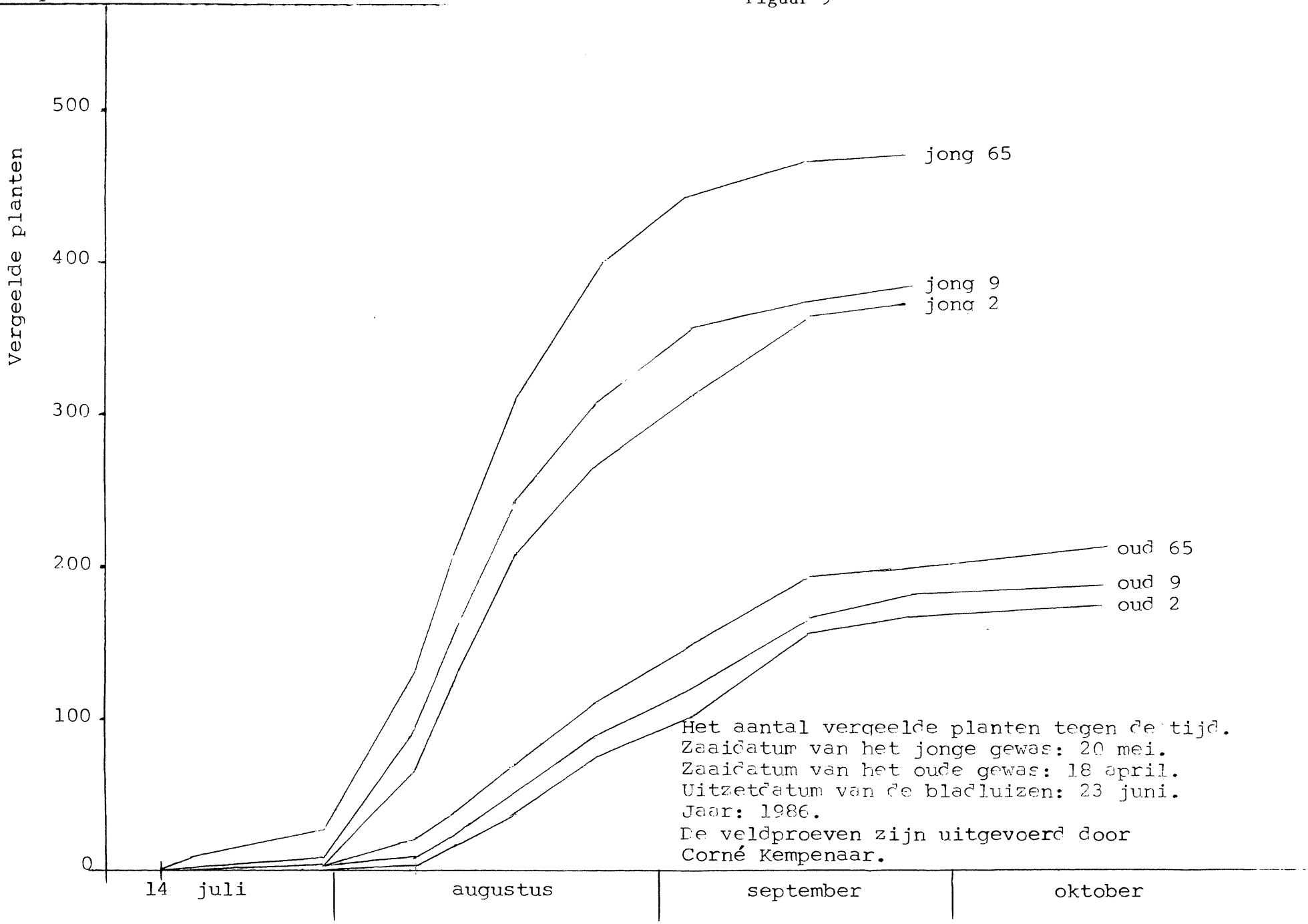
Efficiëntie:
Het aantal vergeelde planten per infectieuze bladluisdag is een maat voor de efficiëntie waarmee de bladluizen planten kunnen infecteren. Samen met het aantal infectieuze bladluisdagen bepaalt deze efficiëntie het aantal planten dat geïnfecteerd zal worden. De invloed van de factoren op de mate van efficiëntie, wordt in aparte grafieken weergegeven. Van elke simulatie wordt het aantal vergeelde planten uitgezet tegen de tijd.

4.2 Model- en veldproefresultaten vergeleken

In de figuur "Veldproefresultaten Minderhoud Hoeve" zijn de resultaten van de veldproeven van Corné Kempenaar uitgezet. De figuur geeft een beeld van het verloop van het aantal vergeelde bieteplanten in het veld gedurende het groeiseizoen. De uitgezette waarden zijn gemiddelden van vier herhalingen. Zonder uitzondering is het aantal vergeelde planten in jong gewas hoger dan in oud gewas. Het aantal adulte bladluizen dat in het veld uitgezet is, heeft weinig invloed op het verloop van de grafieken.
Met het model zijn simulaties gedaan waarbij zoveel mogelijk de veldproefomstandigheden nagebootst werden. Het gesimuleerde verloop van het aantal vergeelde planten in het veld is uitgezet in de figuur "Simulaties met de standaard opstartwaarden".

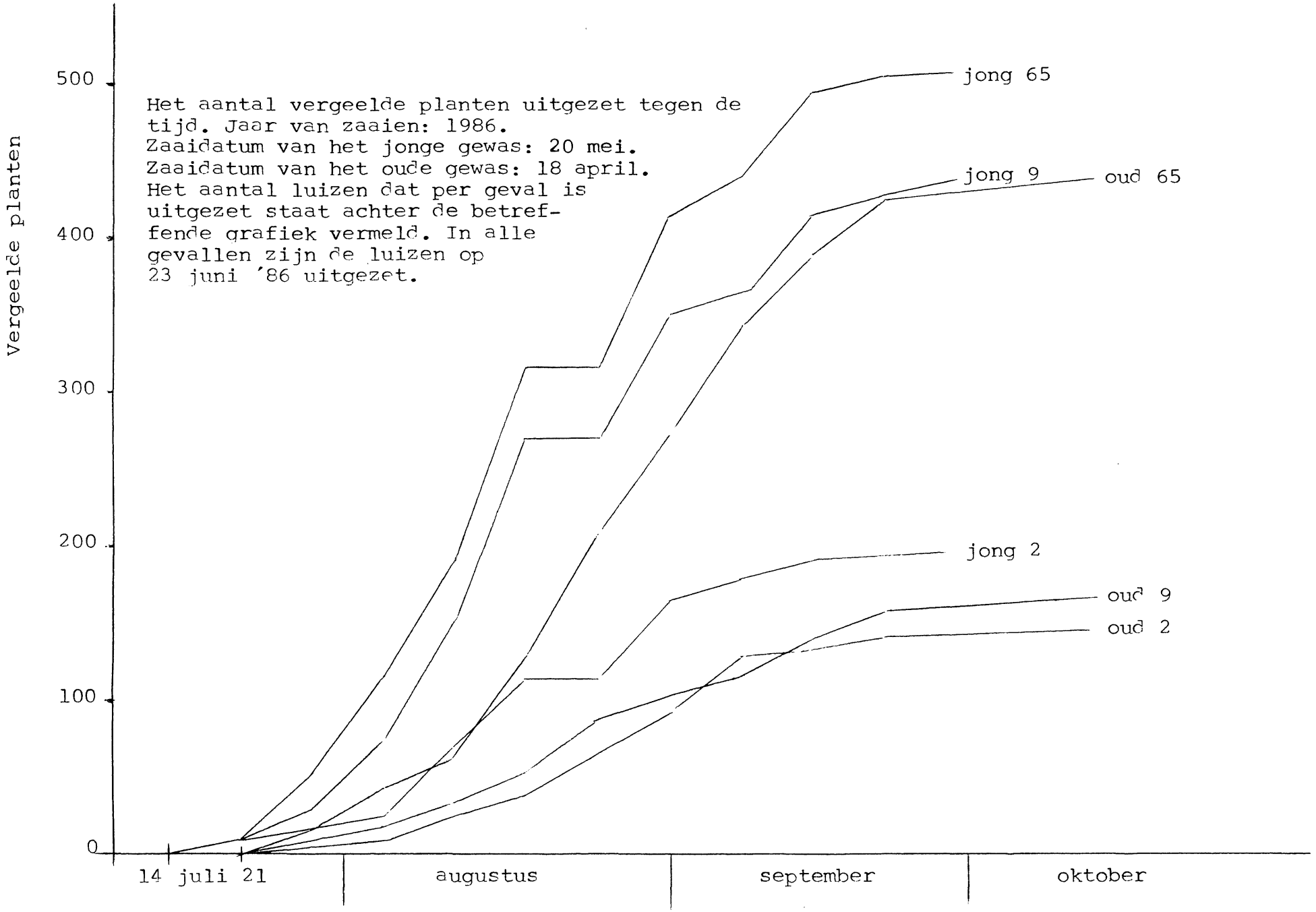
Veldproefresultaten Minderhoud Hoeve

Figuur 9



Simulaties met de standaard opstartwaarden

Figuur 10a



Voor de simulaties zijn de gemiddelden van de getelde aantallen bladluizen, van de vier qua opzet gelijke veldproeven, gebruikt.

Er zijn ook simulaties gedaan met de veldtellingen van vier afzonderlijke veldjes. Het gemiddelde van de resultaten van deze simulaties had vrijwel hetzelfde verloop van vergeelde planten in de tijd. De resultaten van de afzonderlijke simulaties echter kwamen slecht overeen met de resultaten van de bijbehorende veldproeven. De oorzaak hiervan moet gezocht worden in de grote verschillen in getelde bladluizenaantallen in de verschillende veldproeven en de grote invloed die het aantal bladluizen in het model heeft op het aantal vergeelde planten.
In de afzonderlijke veldproeven van Corné Kempenaar kan het voorkomen dat in een veldje waar niet veel bladluizen geteld zijn wel veel vergeling voorkomt en vice versa. Dit komt bij simulaties met dit model nooit voor. Mede hierdoor komen simulaties met gemiddelde waarden van vier bladluizentellingen en het aantal vergeelde planten in het veld beter overeen dan de resultaten van simulaties met afzonderlijke bladluizentellingen.
De resultaten van de simulaties die het meest afwijken van de bijbehorende veldproeven (zie figuur "Simulaties met de standaard opstartwaarden") zijn dan ook van veldsituaties met óf veel getelde bladluizen en weinig vergeling (oud 65) óf weinig getelde bladluizen en veel vergeling (jong 2).

4.3 Gevoeligheidsanalyse

- Bladluizentellingen van de veldjes met de vroeg- en de laatgezaaide bieten omgedraaid

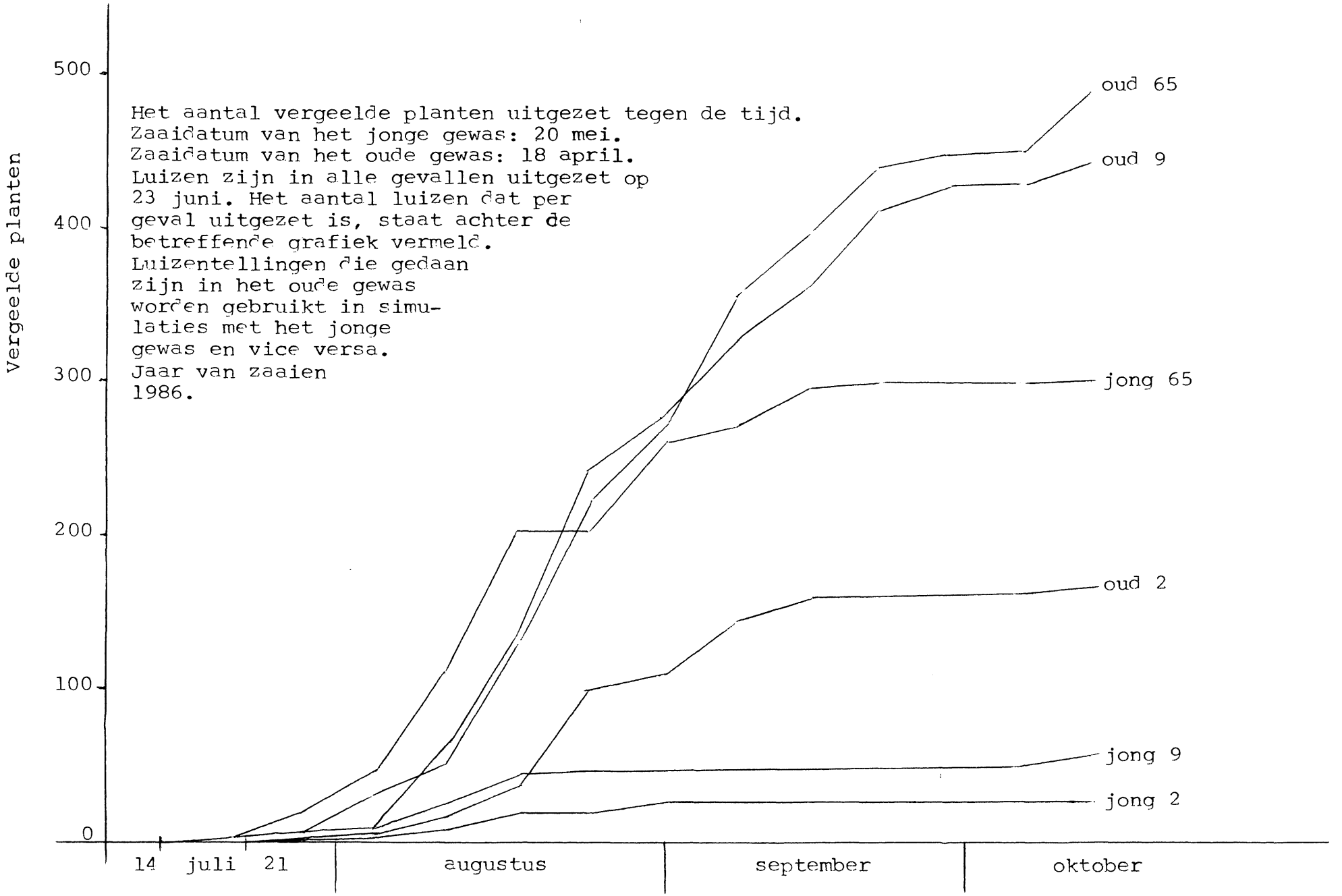
De prestaties (groei en reproductie) van bladluizen liggen bij jonge bieteplanten beduidend hoger dan bij oude planten (Williams, 1987). In veldjes met laatgezaaide bieten worden grotere bladluispopulaties gevonden dan in veldjes met vroeg gezaaide bieten. Ook het aantal vergeelde planten is groter in veldjes met laatgezaaide bieten. Om het effect van de populatiegrootte op het aantal vergeelde planten te bekijken, zijn de bladluistellingen van de vroeggezaaide en de laatgezaaide bieten in de simulaties omgedraaid.

Als de resultaten van deze simulaties vergeleken worden met de resultaten van de simulaties met de standaard opstart-waarden valt in de eerste plaats op dat nu de vergeling in veldjes met vroeggezaaide bieten het grootst is.

Vergeling in de simulaties met laatgezaaide bieten:

Het aantal vergeelde planten is, bij de laatgezaaide bieten met de bladluizentellingen van de vroeggezaaide bieten, lager dan bij de "gewone" simulaties met vroeggezaaide bieten. Dit kan verklaard worden door het feit dat het totale aantal bladluisdagen in deze simulaties lager blijft. Het percentage infectieuze bladluizen ligt gemiddeld rond de 48 procent. Dit is 2 tot 3 keer zo hoog als het gemiddelde percentage infectieuze bladluizen in de simulaties met vroeggezaaide bieten (18%). De oorzaak van het lage aantal

Luizentellingen van het oude en het jonge gewas omgedraaid Figuur 10b



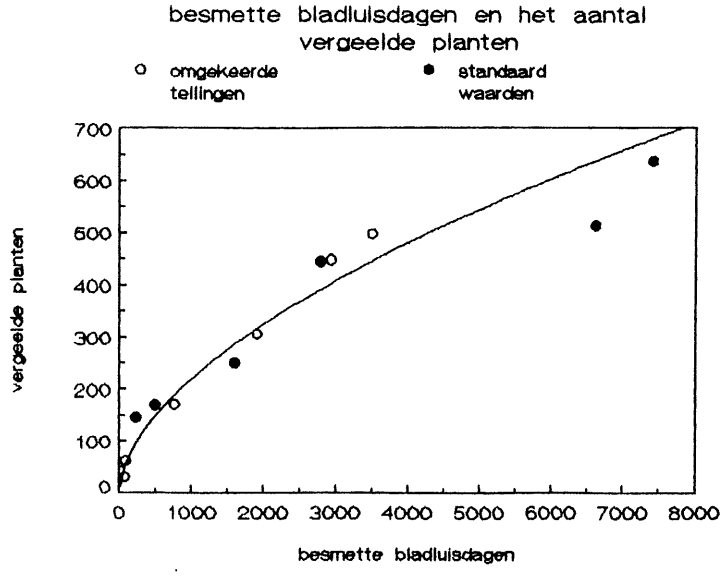
infectieuze bladluisdagen ligt dus in het lage aantal bladluizen.

Vergeling in de simulaties met vroeggezaaide bieten:

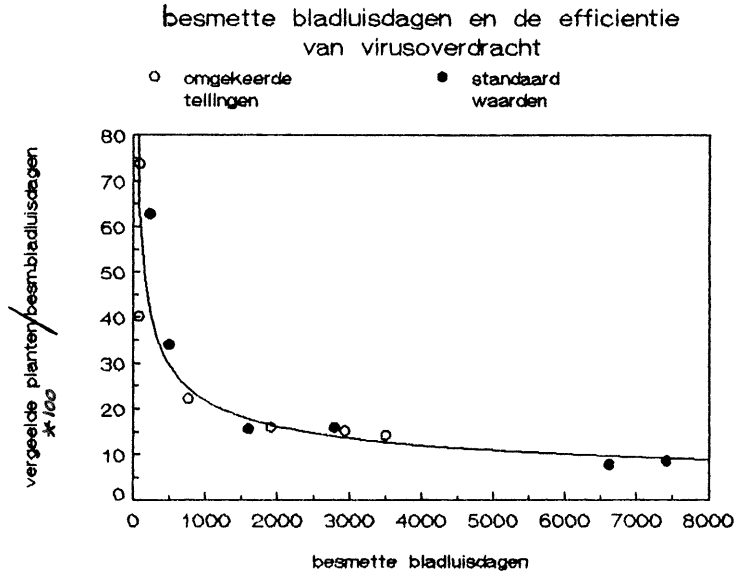
Het aantal vergeelde planten in de simulaties met vroeggezaaide bieten, met de bladluizentellingen van de laatgezaaide bieten, valt ook lager uit dan in de "gewone" simulaties. Aangezien het aantal bladluizen nu zelfs iets hoger is dan in de laatgezaaide bieten (de bladluizen hebben zich vanaf het begin kunnen verspreiden), moet dit het gevolg zijn van een lager percentage infectieuze bladluizen. Het gemiddelde percentage infectieuze bladluizen is inderdaad 2 tot 3 keer zo laag als in simulaties met het laatgezaaide gewas (18% tegen 48%). Hierdoor blijft het aantal infectieuze bladluisdagen en daarmee de vergeling in het veld lager dan in de "gewone" simulaties.

De resultaten van deze simulaties en de resultaten van de simulaties met de standaard opstartwaarden zijn samen in een drietal grafieken verwerkt. In grafiek 11 zijn het aantal infectieuze bladluisdagen uitgezet tegen het aantal vergeelde bieteplanten. Het resultaat is een aanvankelijk steile curve die afvlakt naarmate het aantal infectieuze bladluisdagen toe neemt. Deze verminderde meeropbrengst laat zich als volgt verklaren. Hoe meer infectieuze bladluizen zich gelijktijdig in het veld bevinden, hoe groter de kans dat er meerdere infectieuze bladluizen op één plant zitten. Eén van die bladluizen kan de plant slechts infecteren, de andere bladluizen kunnen daarna niets meer uitrichten.

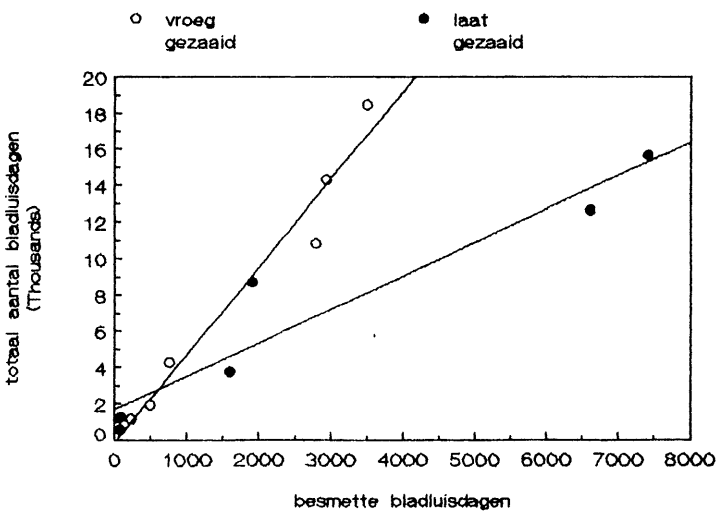
Figuur 11



Figuur 12



Figuur 13



Een benaderingsformule die het verband tussen het aantal vergeelde planten (y) en het aantal infectieuze bladluisdagen (x) geeft:

$y = 4.37x^{0.57}$ (R waarde: 0.98).

Grafiek 12. Het aantal vergeelde planten per infectieuze bladluisdag (y, maat voor efficiëntie) is hier uitgezet tegen het totale aantal infectieuze bladluisdagen (x). Hoe lager het aantal infectieuze bladluisdagen hoe hoger de snelheid waarmee een infectieuze bladluis planten kan infecteren. Deze lijn wordt benaderd met:

$y = 4.37x^{-0.43}$ (R waarde: 0.98).

Grafiek 13. Het aantal infectieuze bladluizen uitgezet tegen het totale aantal bladluizen in de simulatie. In laatgezaaide bieten is het gemiddelde percentage infectieuze bladluizen beduidend hoger dan in vroeggezaaide bieten (48% tegen 18%).

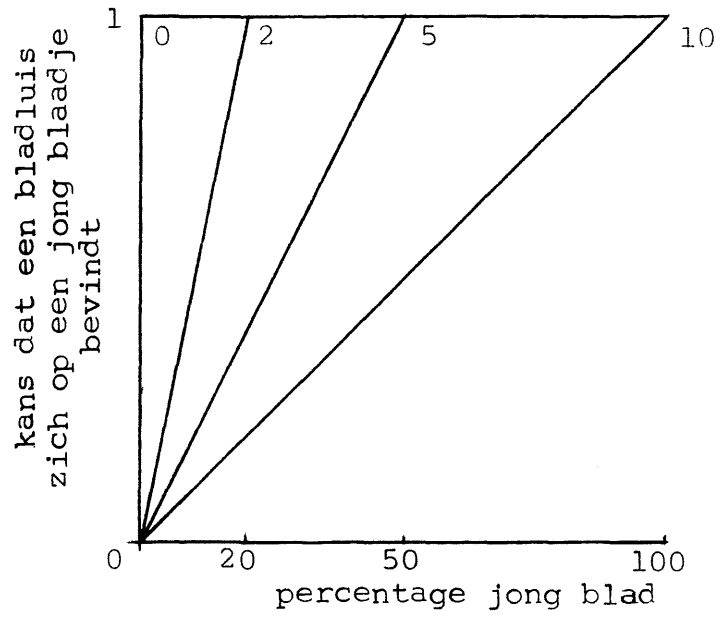
- De rest van factoren waarvoor een gevoeligheidsanalyse gedaan is:

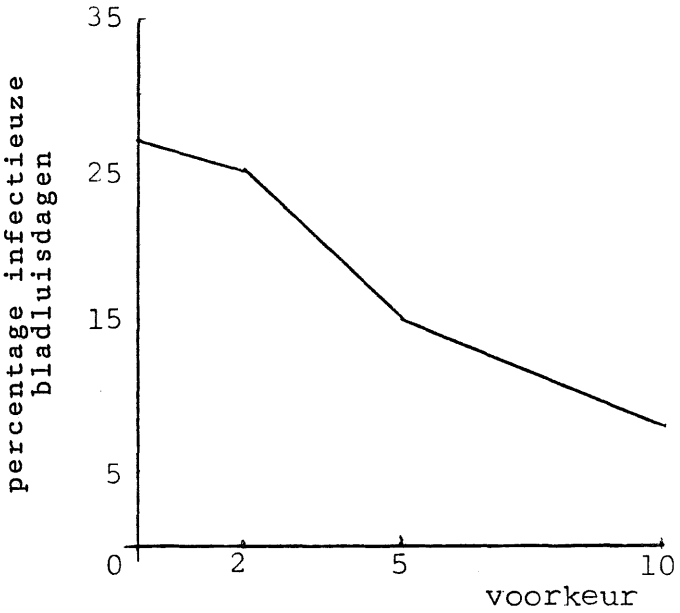
Alleen de resultaten van de simulaties waar negen adulte bladluizen uitgezet werden in veldjes met laatgezaaide bieten, zijn voor de grafieken gebruikt. De resultaten van simulaties met vroeggezaaide bieten en andere aantallen uitgezette bladluizen vertonen dezelfde tendensen. In de grafieken waar het aantal vergeelde planten per infectieuze bladluisdag uitgezet is, is telkens met een stippellijn de verwachte efficiëntie aangegeven. In alle grafieken waarin de efficiëntie is uitgezet, zijn de waarden met een factor 100 vermenigvuldigd.

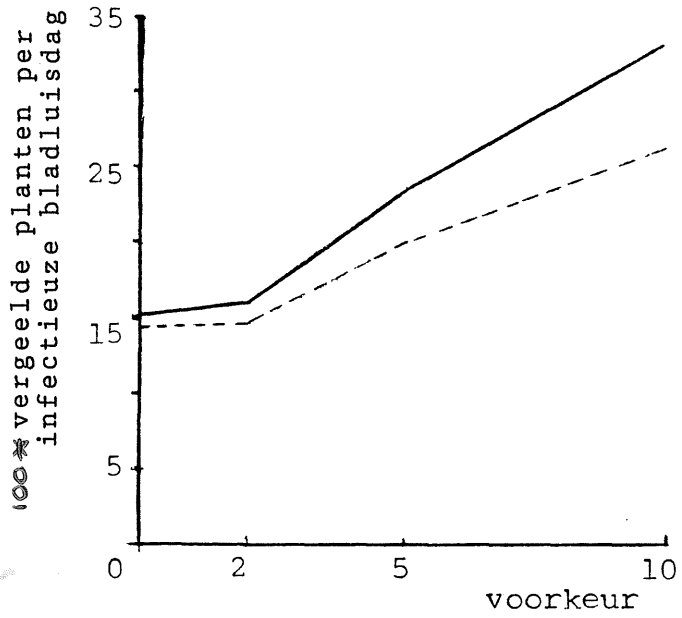
- Voorkeur voor jonge bladeren

Hoe hoger de voorkeur voor jong blad, hoe groter het aantal vergeelde planten. Het aantal bladluizen in de simulatie is telkens gelijk. Het percentage infectieuze bladluizen wordt groter met toenemende voorkeur. Het aantal infectieuze bladluisdagen neemt daardoor toe met een toenemende voorkeur van de bladluizen voor jonge bladeren. De efficiëntie waarmee de bladluizen planten kunnen infecteren blijft ongeveer gelijk. Het hogere aantal vergeelde planten moet volledig toegeschreven worden aan de invloed, die de voorkeur van bladluizen heeft, op het percentage infectieuze bladluizen.

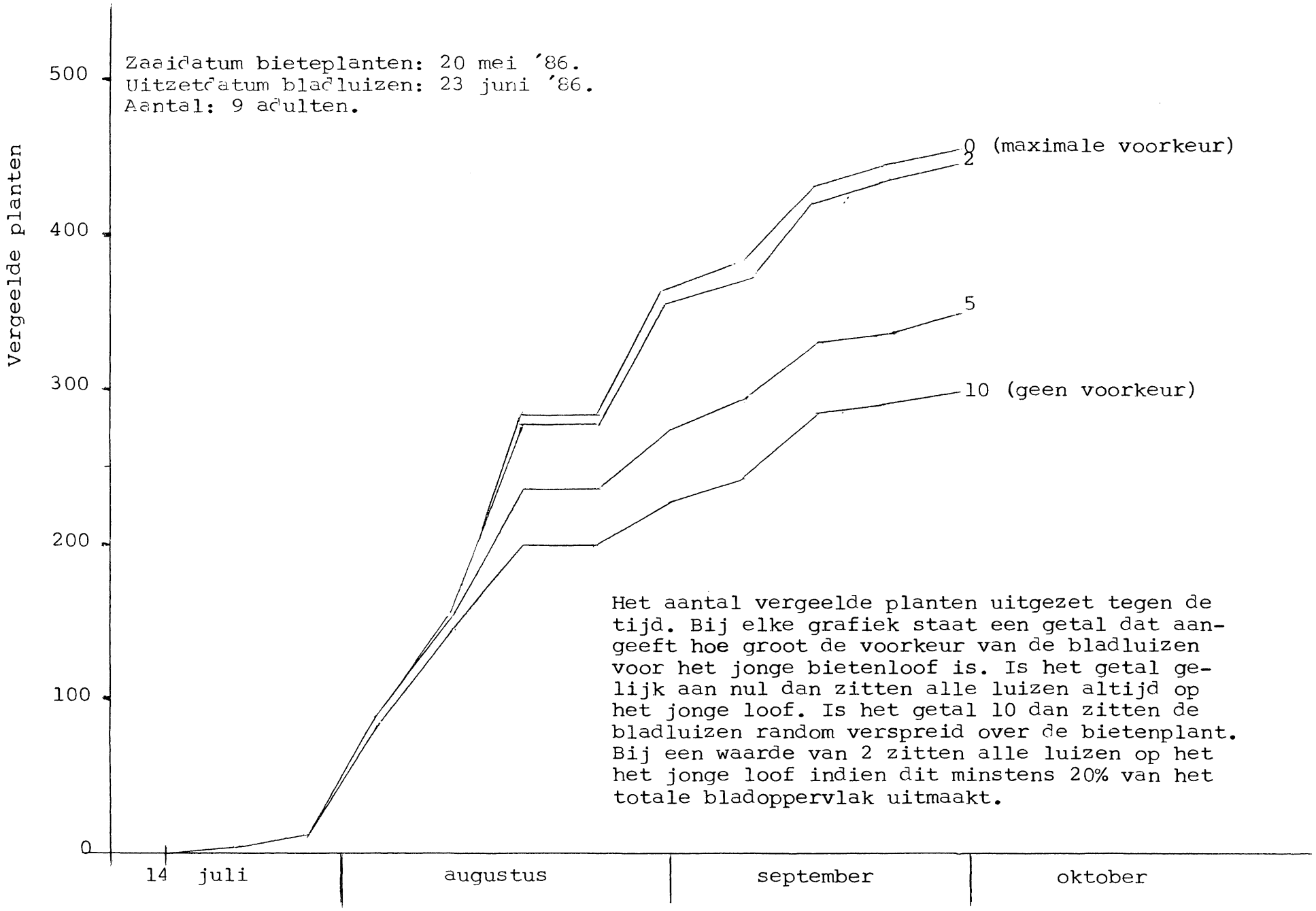
Voorkeur voor jonge bladeren







Voorkeur voor jong blad

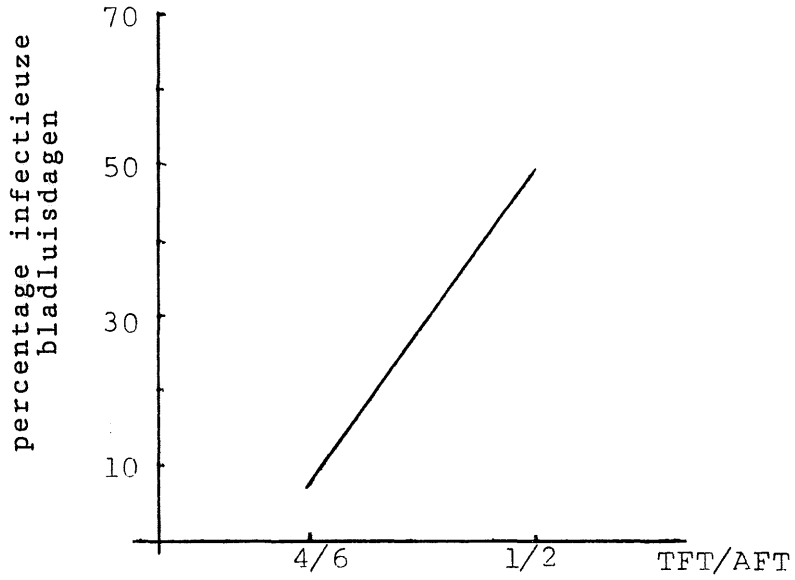


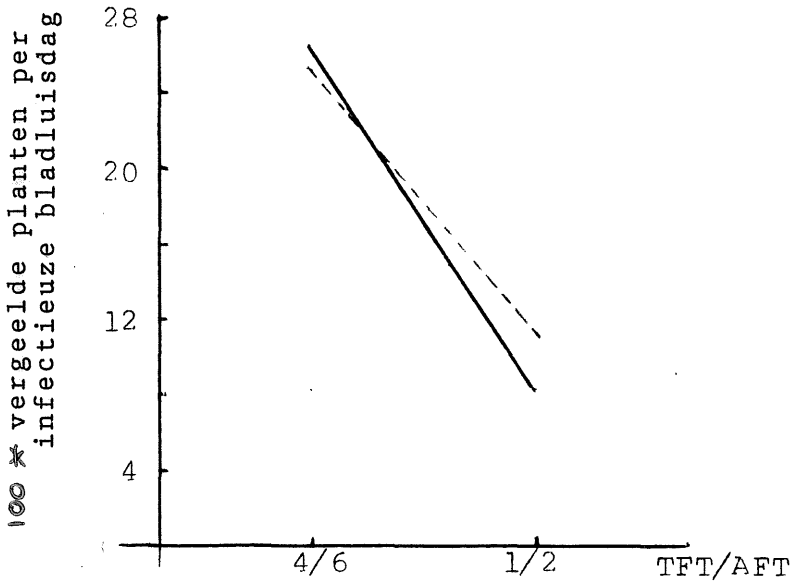
- AFP en TFP

AFP staat voor Aquisition Feeding Period en TFP staat voor Transmission Feeding Period, respectievelijk de acquisitie- en afgifteperiode van het virus door de bladluizen.

Hoe korter AFP en TFP, hoe groter het aantal vergeelde planten in het veld. Het aantal bladluizen is telkens gelijk. Alleen het percentage infectieuze bladluizen wordt groter naarmate AFP en TFP kleiner worden. Het grotere aantal infectieuze bladluisdagen is dus het resultaat van het hogere percentage infectieuze bladluizen. De efficiëntie waarmee de bladluizen het virus overdragen verandert nauwelijks. In de simulaties met AFP = 6 en TFP = 4 uur zijn er 756 infectieuze bladluisdagen. Onder standaardcondities zou dit een snelheid van .25 planten per infectieuze bladluisdagen zijn. Er wordt een snelheid van .26 planten per infectieuze bladluisdagen gevonden. Bij AFP = 2 en TFP = 1 uur is de verwachte efficiëntie .08 en de gevonden waarde .11.

Efficiëntie van virusoverdracht





Efficiëntie van virusoverdracht

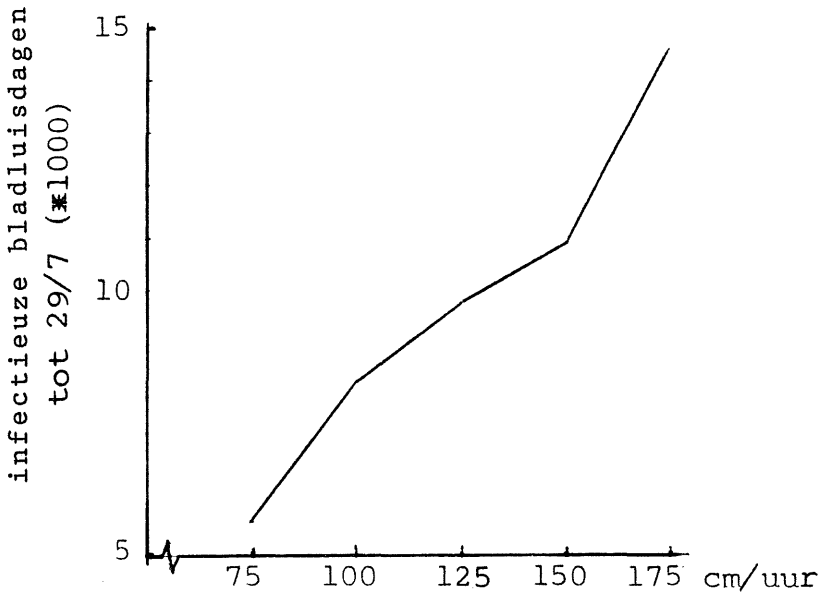
Het aantal vergeelde planten uitgezet tegen de tijd. Bij elke grafiek staan twee getallen. Het bovenste getal geeft de opnametijd van het virus aan in uren. Het onderste getal geeft de afgiftetijd aan in uren. Zaaidatum bieten: 20 mei '86. Uitzetdatum luizen: 23 juni. (negen adulten)

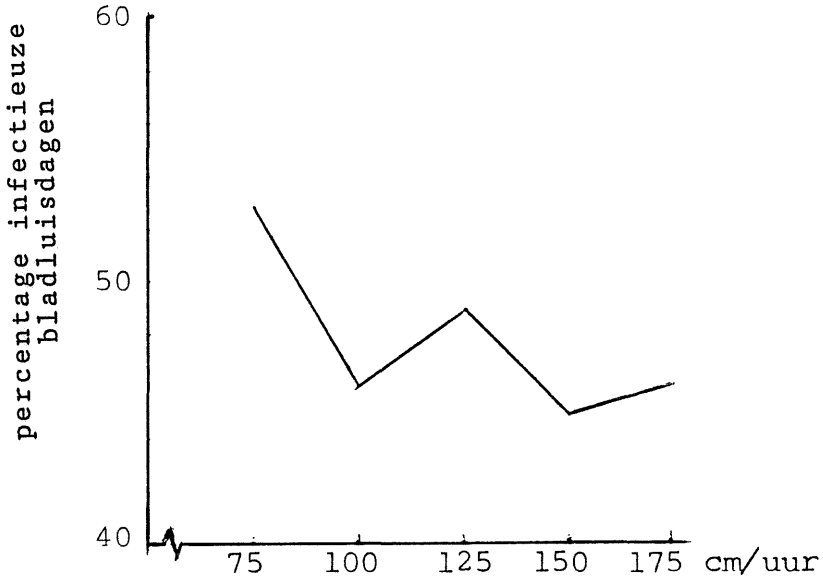
- Maximum loopsnelheid van de bladluizen

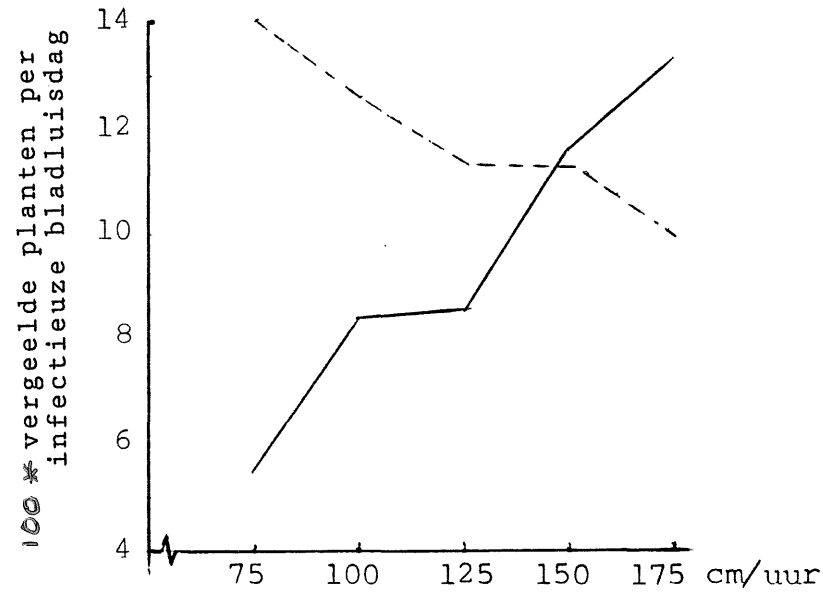
Met maximum loopsnelheid wordt bedoeld de afstand in één bepaalde richting die de bladluis kan afleggen per uur onder optimale omstandigheden. Voor het model wil dat zeggen, bij een leeftijd van 25 dagen en een temperatuur van 30 graden Celcius.

Hoe groter de loopsnelheid, hoe groter het aantal vergeelde planten. Het aantal bladluizen in de simulatie neemt toe met toenemende max. loopsnelheid. Het percentage infectieuze bladluizen is vrijwel constant. Er lijkt een lichte afname te zijn bij hoge max. loopsnelheden. De efficiëntie van virusoverdracht neemt toe met toenemende en af met afnemende max. loopsnelheid. Het grotere aantal vergeelde planten bij hoge max. loopsnelheid is het gevolg van èn een toegenomen efficiëntie èn een hoger aantal infectieuze bladluisdagen. Dit hogere aantal infectieuze bladluisdagen is voornamelijk het gevolg van de hogere aantallen bladluizen in de simulaties. Bij max. loopsnelheid van 75 cm/uur is de verwachte efficiëntie .14 en de gevonden efficiëntie .05. Bij max. loopsnelheid van 175 cm/uur respectievelijk .13 en .10. Vooral bij lage max. loopsnelheid is de gevonden efficiëntie lager dan de verwachte efficiëntie.

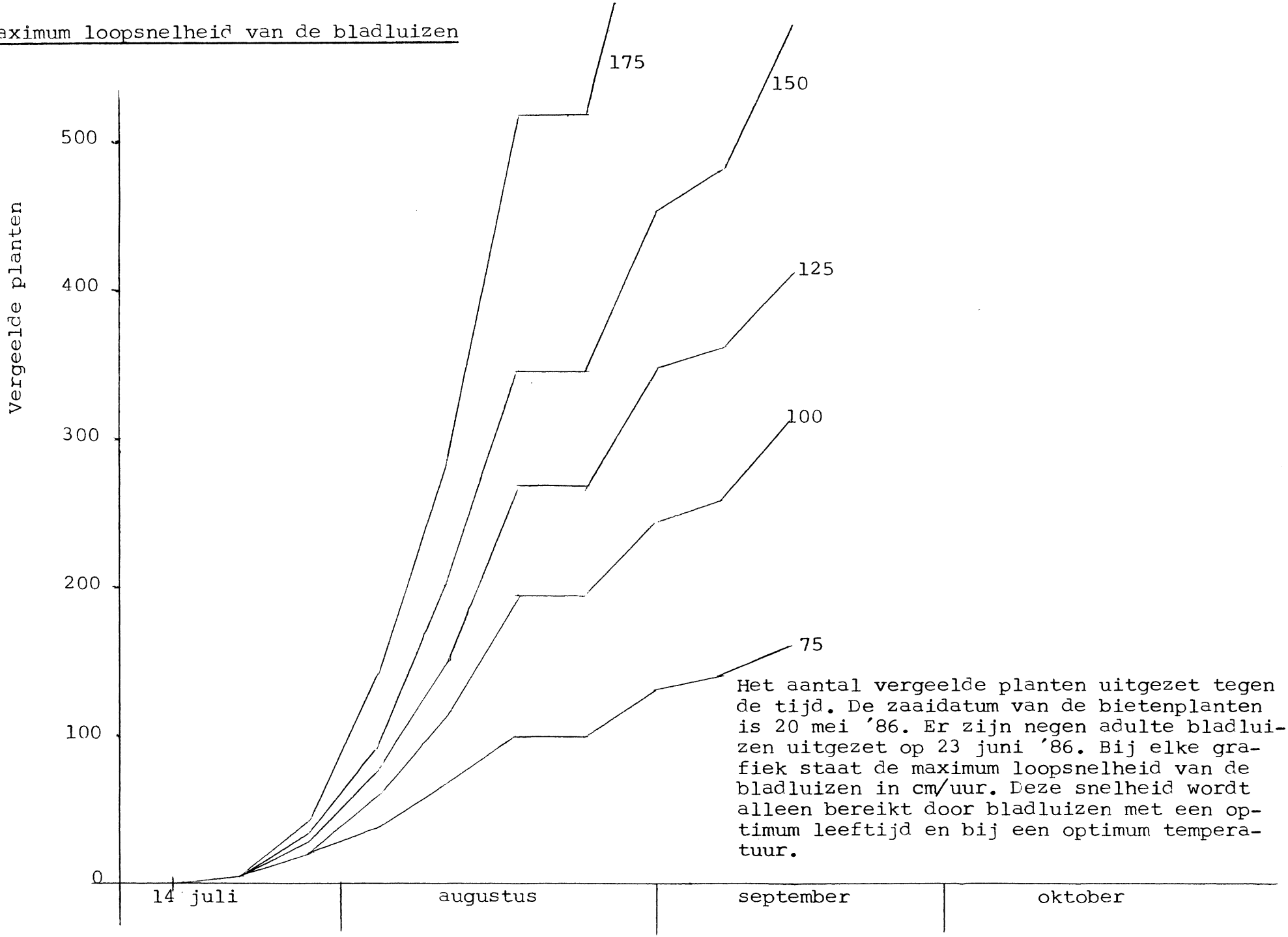
Maximum loopsnelheid







Maximum loopsnelheid van de bladluizen

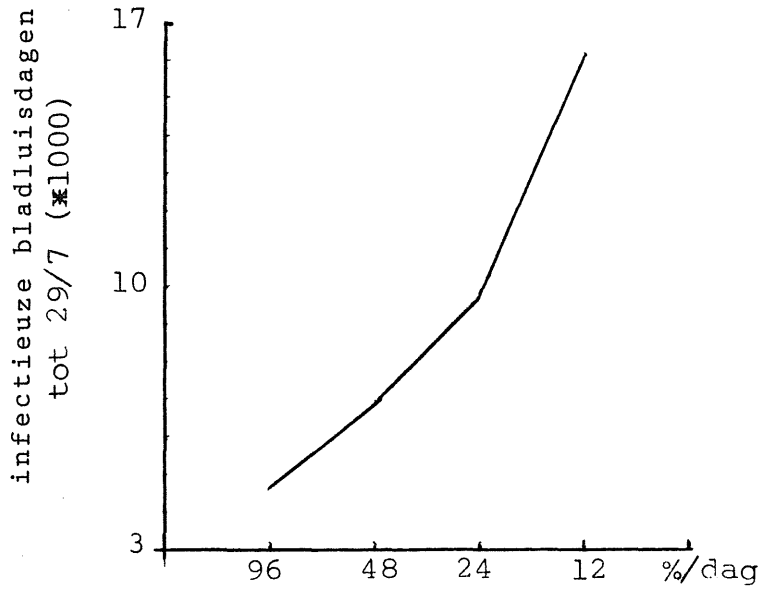


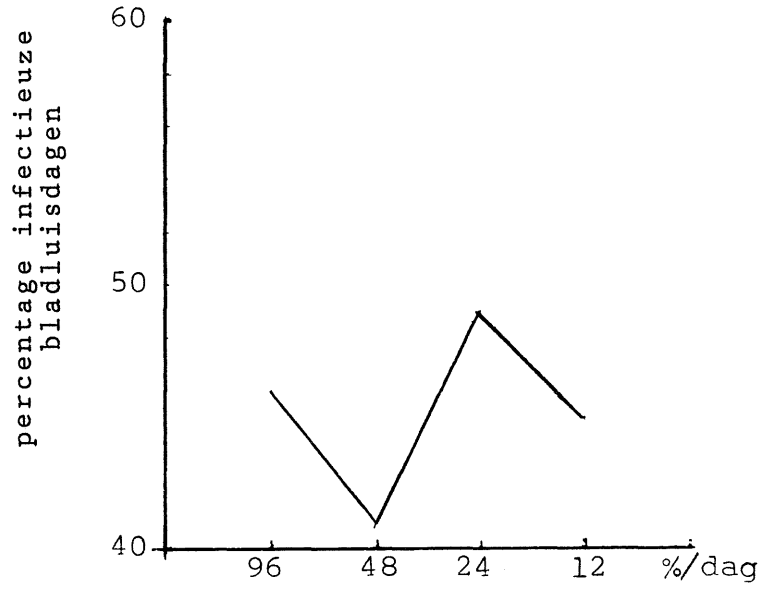
Het aantal vergeelde planten uitgezet tegen de tijd. De zaaidatum van de bietenplanten is 20 mei '86. Er zijn negen adulte bladluizen uitgezet op 23 juni '86. Bij elke grafiek staat de maximum loopsnelheid van de bladluizen in cm/uur. Deze snelheid wordt alleen bereikt door bladluizen met een optimum leeftijd en bij een optimum temperatuur.

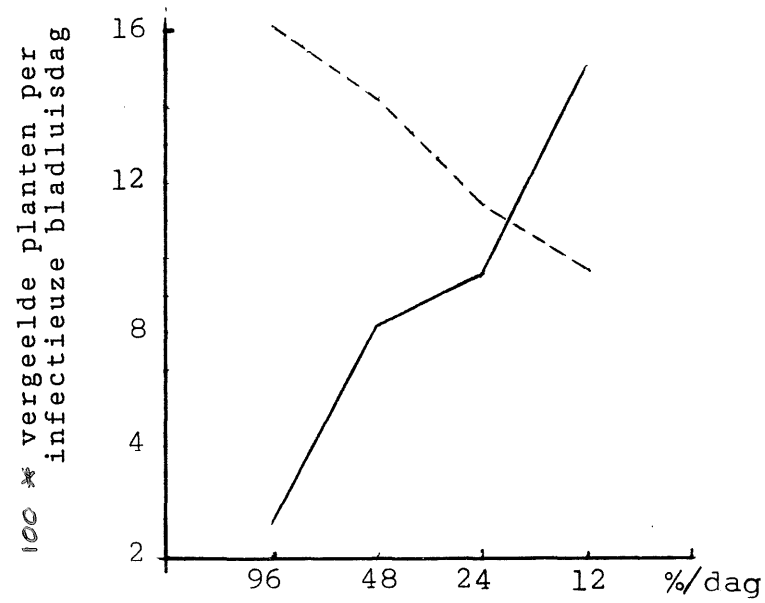
- Predatie

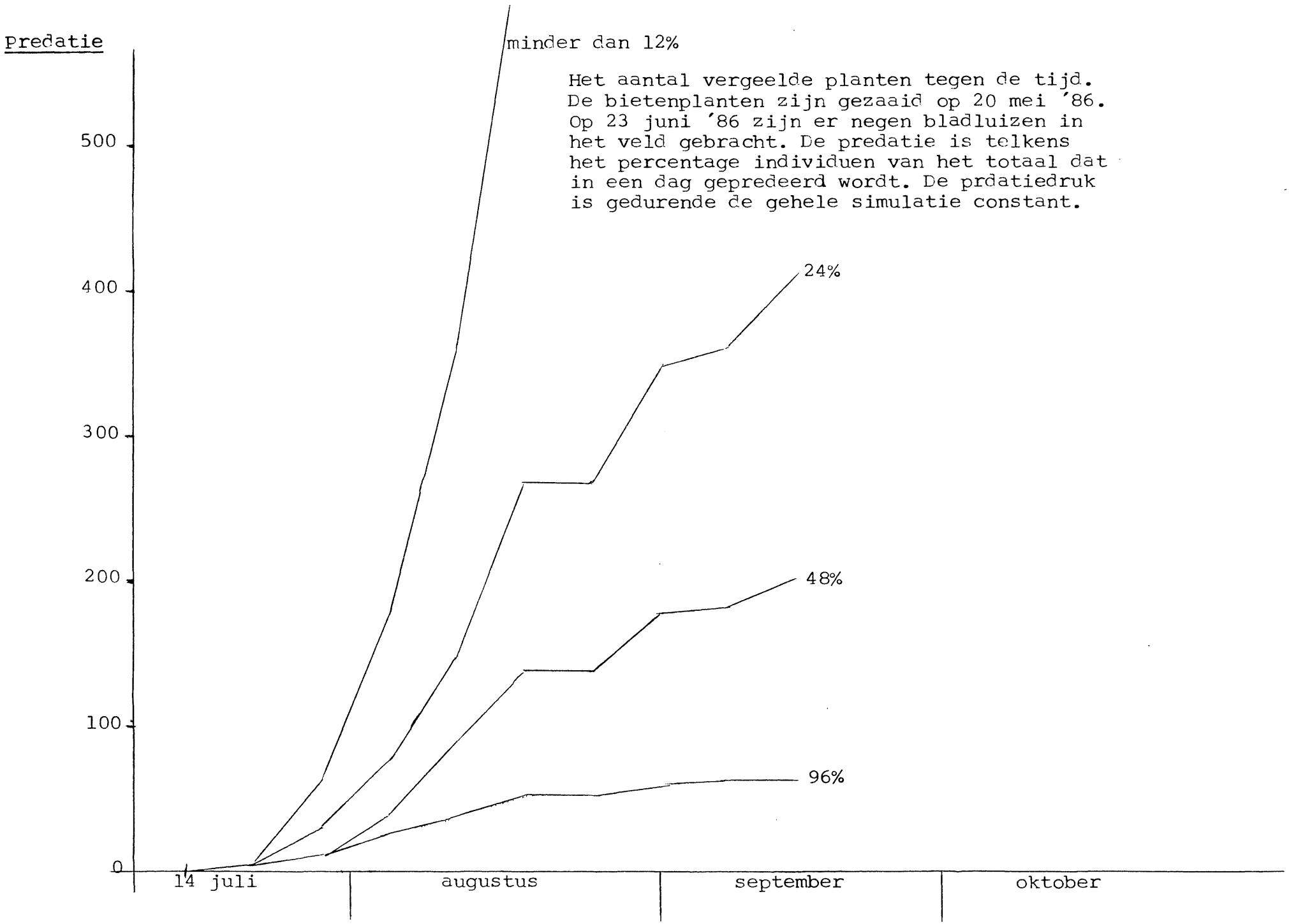
Hoe hoger de predatiedruk, hoe lager het aantal vergeelde planten in de simulaties. De hoogte van de predatiedruk heeft invloed op het aantal bladluizen in de simulatie en niet tot nauwelijks op het percentage infectieuze bladluizen. Het hogere aantal infectieuze bladluisdagen is hier vooral een gevolg van de hogere bladluisaantallen. De efficiëntie van virusoverdracht neemt af bij toenemende predatiedruk. Bij een predatiedruk van 96% van het aantal bladluizen per uur is de efficiëntie .03 terwijl een efficiëntie van .16 verwacht werd. Bij een predatiedruk van 12% van het aantal bladluizen per uur respectievelijk .15 en .09. Het hoge aantal vergeelde planten bij lage predatiedruk is het gevolg van de toegenomen efficiëntie en het hoge aantal infectieuze bladluisdagen.

Percentage gepredeerde bladluizen per dag







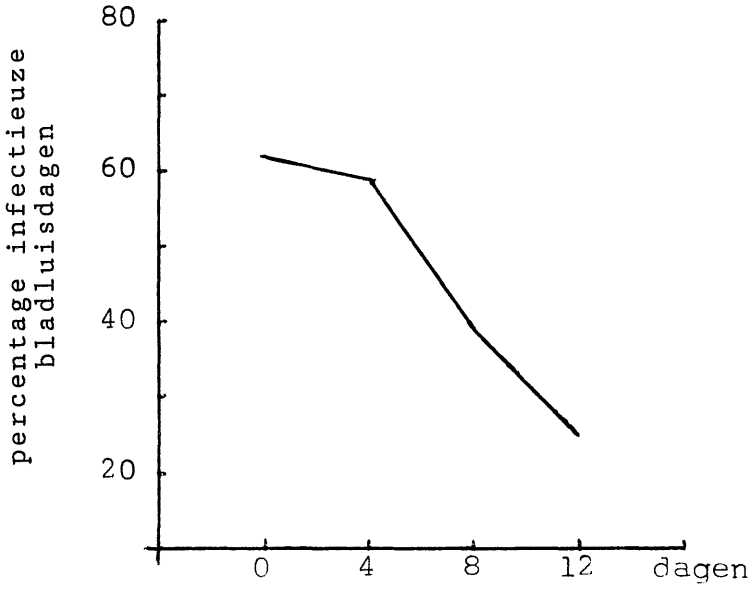

Predatie
minder dan 12%
Het aantal vergeelde planten tegen de tijd. De bietenplanten zijn gezaaid op 20 mei '86. Op 23 juni '86 zijn er negen bladluizen in het veld gebracht. De predatie is telkens het percentage individuen van het totaal dat in een dag gepredeerd wordt. De prdatiedruk is gedurende de gehele simulatie constant.
500
400
300
200
100
0
24%
48%
96%
14 juli
augustus
september
oktober

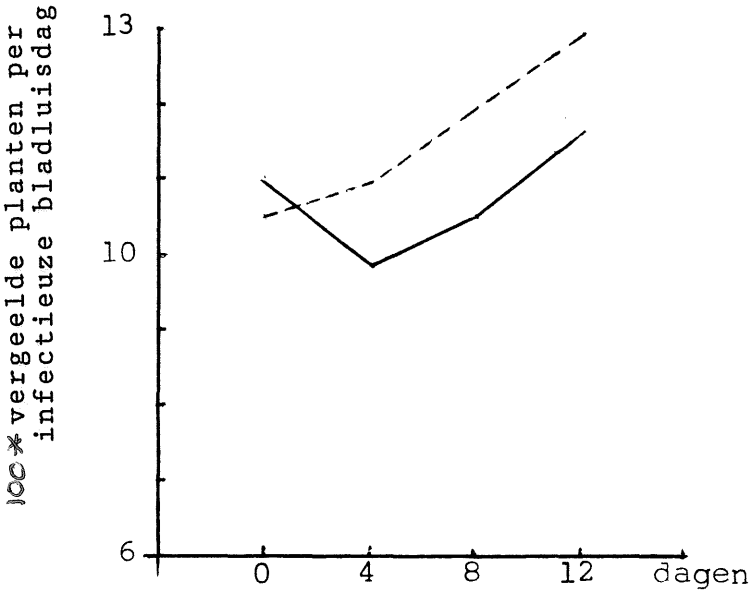
- Latentieperiode

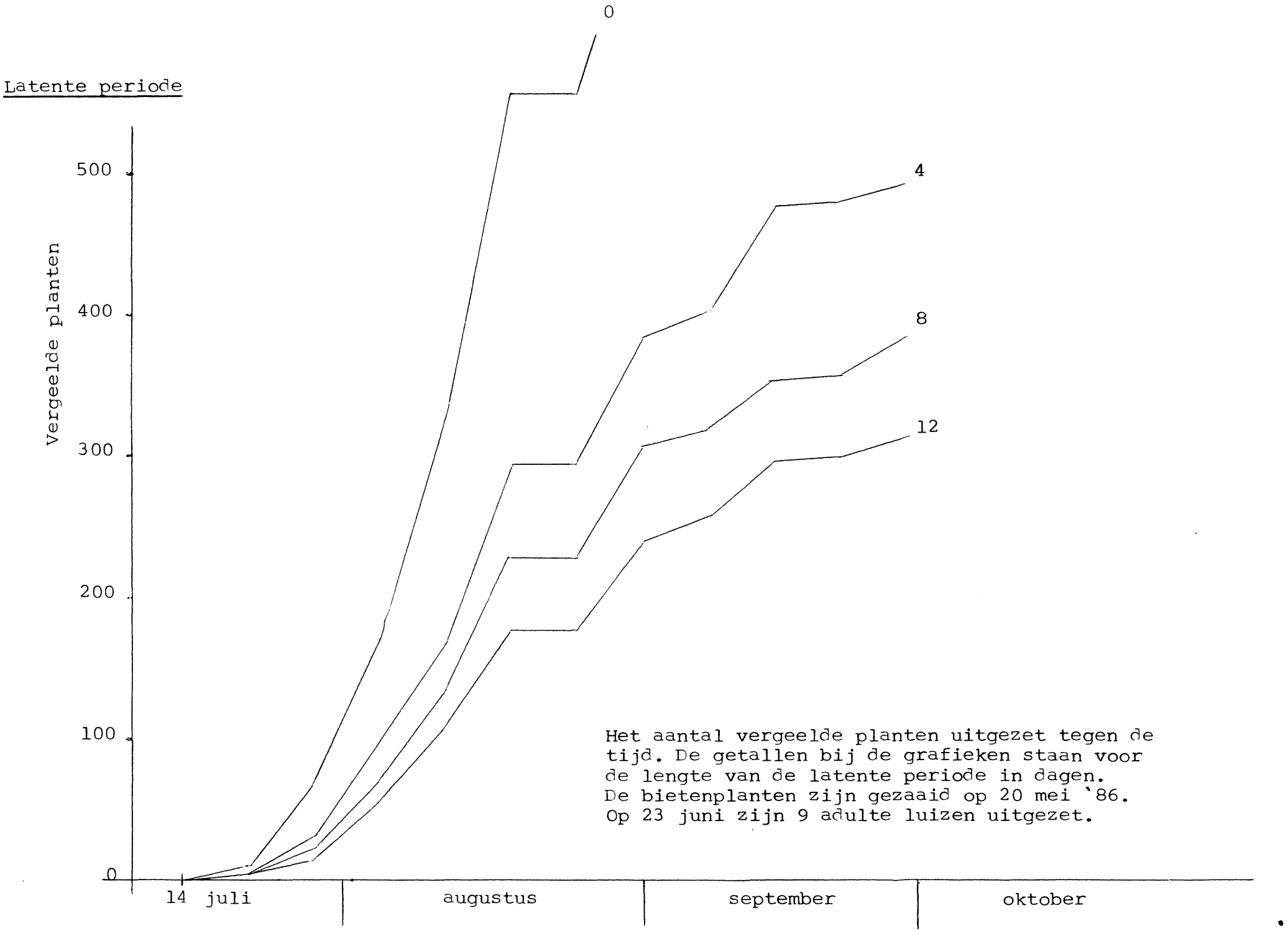
De latentieperiode is gedurende het groeiseizoen niet constant. In het model neemt de latentieperiode toe van 4 tot 12 dagen. Voor de gevoeligheidsanalyse is de latentieperiode gedurende het gehele seizoen op een constante waarde gezet.

Hoe korter de latentieperiode, hoe hoger het aantal vergeelde planten in de simulatie. Het percentage infectieuze bladluizen neemt toe bij kleinere latentieperiodes. Het aantal bladluizen in de simulaties is telkens gelijk. Variaties in het aantal infectieuze bladluisdagen worden dus veroorzaakt door variaties in het percentage infectieuze bladluizen. De efficiëntie waarmee de bladluizen het virus verspreiden blijft vrijwel gelijk aan de verwachte efficiëntie. Bij een latentieperiode van 0 dagen is de verwachte efficiëntie .09 en de gevonden efficiëntie .10. Bij een latentieperiode van 12 dagen respectievelijk .14 en .12. Het grotere aantal vergeelde planten bij een korte latentieperiode is het gevolg van hogere aantallen infectieuze bladluisdagen.

Lengte van de latente periode






Latente periode
Vergeelde planten
500
400
300
200
100
0
14 juli
augustus
september
oktober
0
4
8
12
Het aantal vergeelde planten uitgezet tegen de tijd. De getallen bij de grafieken staan voor de lengte van de latente periode in dagen.
De bietenplanten zijn gezaaid op 20 mei `86.
Op 23 juni zijn 9 adulte luizen uitgezet.

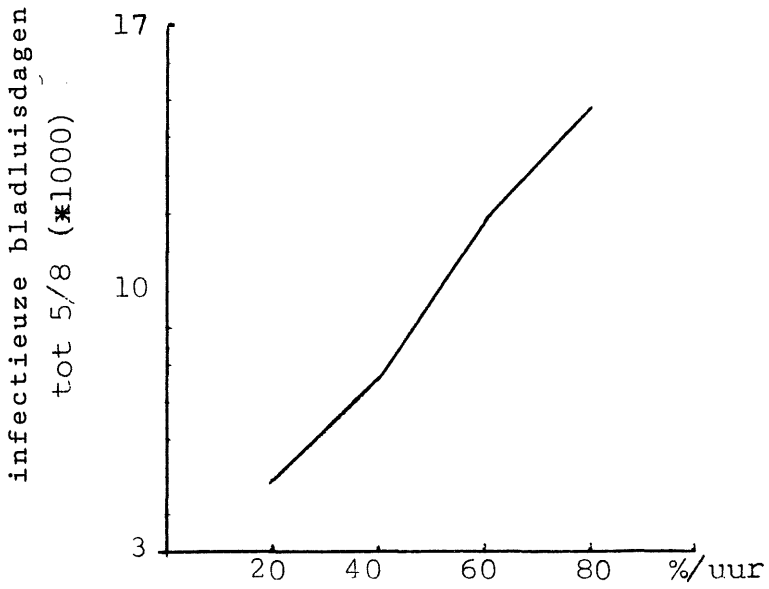
- Percentage lopende bladluizen

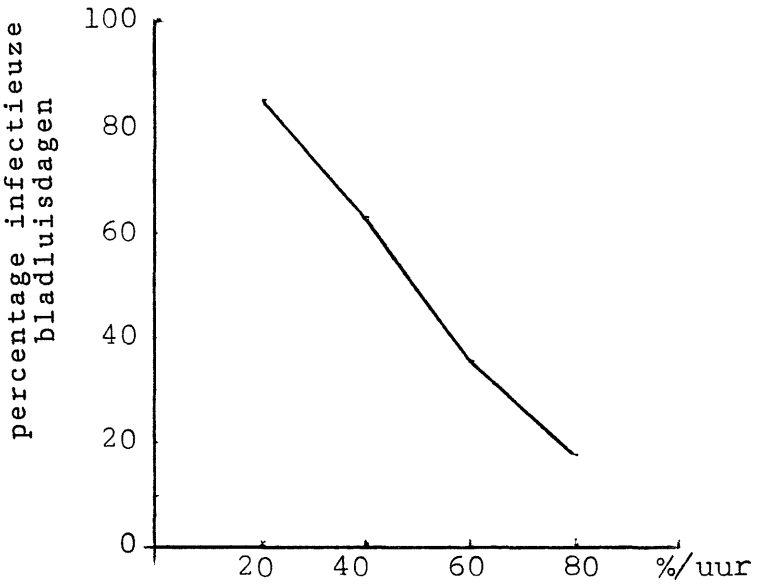
Het percentage bladluizen dat loopt binnen één uur is niet constant gedurende het groeiseizoen. Voor de gevoeligheids-analyse is het percentage telkens wel constant gehouden.

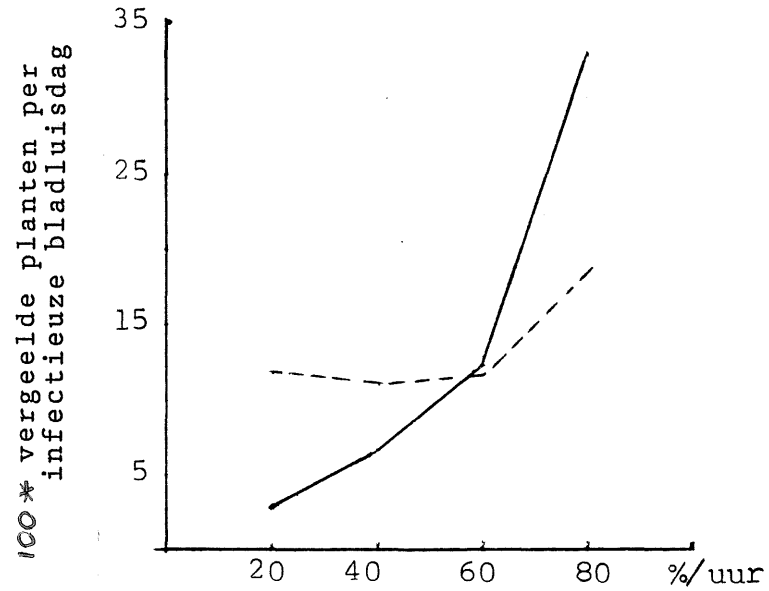
Het aantal vergeelde planten neemt aanvankelijk toe met het aantal lopende bladluizen per uur. Rond het punt waar 60% bladluizen per uur lopen is een optimum bereikt in de vergeling in het veld. Het aantal bladluizen in de simulatie neemt toe met het percentage lopende bladluizen per uur. Het percentage infectieuze bladluizen daalt hierbij zo sterk, dat het aantal infectieuze bladluisdagen in de simulatie daalt ondanks het toenemende aantal bladluizen. De efficiëntie waarmee bladluizen planten kunnen infecteren neemt toe met een toenemend percentage lopers per uur. Bij een percentage lopers van 20% per uur is de verwachte efficiëntie .12 en de gevonden efficiëntie .03. Bij een percentage lopers van 80% per uur respectievelijk .19 en .30. Het lage aantal vergeelde planten bij lage percentages lopende bladluizen per uur wordt vooral veroorzaakt door een lage efficiëntie van virusoverdracht. Bij hoge percentages lopende bladluizen neemt het aantal vergeelde planten af door de lage percentages infectieuze bladluizen. Het verband tussen het percentage infectieuze bladluizen (y) en het percentage lopers per uur (x) wordt grofweg door de volgende formule beschreven:

$$y = 100 - x$$

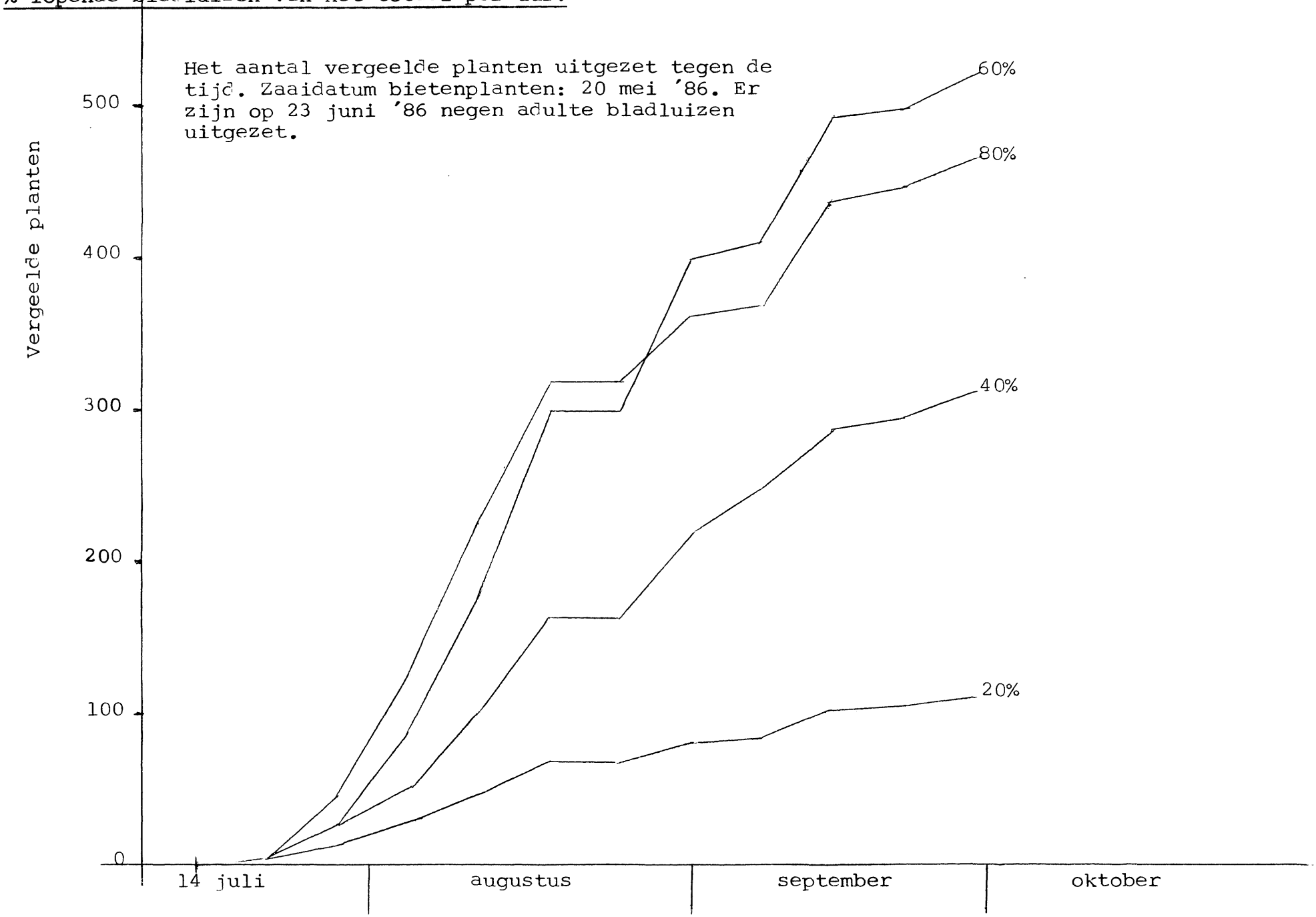
Percentage lopende luizen per uur







% lopende bladluizen van het totaal per uur.



De verwachte efficiëntie van virusoverdracht is steeds gebaseerd op het aantal infectieuze bladluisdagen in de simulaties en het verband hiervan met het aantal vergeelde planten in het veld onder standaardcondities (zie H4.3; Gevoeligheids analyse - Bladluizentellingen van de veldjes met de vroeg en de laatgezaaide bieten omgedraaid).

Hoofdstuk 5. Discussie

Wat veroorzaakt het verschil in verspreidingssnelheid van BYV in veldjes met vroeg- en met laatgezaaide bieten? Uit de gevoeligheidsanalyses is gebleken dat deze vraag voor het model tot de volgende vragen herleid kan worden:

-1- wat veroorzaakt een verschil in verspreidingssnelheid van Myzus persicae in veldjes met vroeg- en met laatgezaaide bieten,

-2- hoeveel infectieuze Myzus persicae bevinden zich in de veldjes met vroeg- en in de veldjes met laatgezaaide bieten?

- De verspreidingssnelheid van de bladluizen en de efficiëntie van virusoverdracht (bij -1-)

De verspreidingssnelheid van de bladluizen en het aantal infectieuze bladluisdagen beïnvloeden allebei de efficiëntie waarmee het virus overgedragen wordt. Het verband tussen de efficiëntie van virusoverdracht en het aantal infectieuze bladluisdagen is in simulaties met vroeg- en met laatgezaaide bieten hetzelfde (figuur 12). De verspreidingssnelheid van de bladluizen moet daarom in vroeg- en in laatgezaaide bieten ook gelijk zijn.

De verspreidingssnelheid van de bladluizen hangt af van de loopsnelheid en het loopgedrag van de bladluizen. In het model is de loopsnelheid in vroeg- en in laatgezaaide bieten even groot. De loopsnelheid hangt af van de temperatuur en

de leeftijd van de bladluizen. De temperatuur is in de simulaties met de vroeg- en in de simulaties met de laatgezaaide bieten even hoog. De leeftijd van de bladluizen is gemiddeld even hoog omdat de predatiedruk even hoog is.
Alleen het loopgedrag verschilt. Op jonge bieteplanten is de loopfrequentie van de bladluizen hoger dan op oude bieteplanten. De planten in de vroeg- en in de laatgezaaide veldjes verschillen in leeftijd slechts één maand. Bij dit leeftijdverschil is het verschil in het percentage lopers per uur slechts 4% (zie figuur 4).
De verspreiding van de bladluizen komt in de laatgezaaide bieten later op gang dan in de vroeggezaaide bieten. Dit is het gevolg van het nog niet aaneensluiten van de bladeren in de laatgezaaide bieten op het moment van uitzetten van de bladluizen.
Blijkbaar compenseren het latere op gang komen van de verspreiding en het iets hogere percentage lopers per uur elkaar goeddeels.

- Het aantal infectieuze bladluisdagen (bij -3-).

Het aantal infectieuze bladluisdagen is het product van het aantal infectieuze bladluizen met de periode (in dagen) waarbinnen ze planten hebben geïnfecteerd. In de simulaties is het aantal infectieuze bladluizen in jonge bieteplanten hoger dan in oude bieteplanten. Dit verschil wordt veroorzaakt door èn een hoger aantal bladluizen èn een hoger percentage infectieuze bladluizen. Omdat de efficiëntie van virusoverdracht in veldjes met vroeg- en met laatgezaaide bieten even groot is, moet het verschil in vergeling

veroorzaakt worden door dit verschil in het aantal infectieuze bladluisdagen.
In werkelijkheid worden er in veldjes met jonge bieteplanten inderdaad grotere bladluispopulaties gevonden dan in veldjes met oude planten. Dit is het gevolg van een hogere groei- en reproductiesnelheid van bladluizen op jonge bieteplanten (Williams, 1987).

Hieronder volgt een bespreking van de factoren die invloed kunnen hebben op het aantal vergeelde planten in het veld. De factoren zijn onderverdeeld in drie groepen (A, B en C).

-A- Factoren die het percentage infectieuze bladluizen in de simulaties hebben beïnvloed.

-1- De latentieperiode
De latentieperiode is voor jonge bieten korter dan voor oude bieten. Dit betekent dat in veldjes met laatgezaaide bieten het virus door de bladluizen sneller uit de planten opgenomen kan worden na inoculatie van een plant dan in veldjes met vroeggezaaide bieten. In veldjes met vroeggezaaide bieten zullen er daardoor meer bladluizen zijn die zich op reeds geïnoculeerde planten bevinden waaruit het virus nog niet opneembaar is. Dit heeft een negatief effect op het percentage infectieuze bladluizen. Uit de gevoeligheidsanalyse is gebleken dat het percentage infectieuze bladluizen toeneemt met een afnemende latentieperiode. Omdat de latentieperiode in jonge planten korter is dan in oude planten, draagt dit bij aan het

grotere aantal vergeelde planten in veldjes met laatgezaaide (relatief jonge) bieteplanten.

-2- Het percentage besmet bladoppervlak.

Dat het percentage besmet bladoppervlak in veldjes met jonge bieteplanten sneller toeneemt dan in veldjes met oude bieteplanten wordt geïllustreerd in bijlage A en B. De planten bereiken in de simulatie met laatgezaaide bieten hogere besmettingsklassen. Het maximum bereikbare percentage besmet bladoppervlak ligt, afhankelijk van de leeftijd van de plant op het moment van inoculatie, tussen de 0 en 100%. Planten die geïnoculeerd worden op jonge leeftijd krijgen een groter percentage besmet bladoppervlak dan planten die op late leeftijd geïnoculeerd worden. Hoe groter het percentage besmet bladoppervlak, hoe hoger het percentage bladluizen dat zich op besmet blad bevindt. Hierdoor is het percentage infectieuze bladluizen ook hoger. Dit draagt bij aan de grotere mate van vergeling in veldjes met laatgezaaide planten.

-3- Het percentage bladluizen dat loopt per uur.

Uit de gevoeligheidsanalyse is gebleken dat het percentage bladluizen dat loopt per uur een zeer groot effect heeft op het percentage infectieuze bladluizen in de simulaties. Het verband tussen het percentage infectieuze bladluizen (y) en het percentage lopers per uur (x) wordt grofweg door de volgende formule beschreven:

$$y = 100 - x$$

In simulaties met vroeggezaaide bieten en in simulaties met laatgezaaide bieten verschilt het percentage lopende bladluizen per uur slechts vier procent. Dit verschil verkleint het gemeten verschil van het percentage infectieuze bladluizen in vroeg- en in laatgezaaide bieten (30%) met ongeveer 4%.
Hoe hoger het percentage lopende bladluizen per uur, hoe hoger de efficiëntie van virusoverdracht door de bladluizen. Deze hogere efficiëntie compenseert het lagere percentage infectieuze bladluizen tot een percentage lopende bladluizen van 60% per uur. Hierbij is het aantal vergeelde planten in de simulaties maximaal. In de simulaties met de veldjes met laatgezaaide bieten ligt het percentage lopende bladluizen per uur aldoor dichter bij dit optimum dan in de veldjes met de vroeggezaaide bieten. Mede hierdoor is het aantal vergeelde planten in veldjes met laatgezaaide bieten hoger dan in veldjes met vroeggezaaide bieten.

-B- Uit de gevoeligheidsanalyses zijn een paar factoren naar voren gekomen die tevens invloed hebben op het percentage infectieuze bladluizen in de simulaties maar die in het standaard model constant verondersteld zijn.

-4- De maximum loopsnelheid.
Hoe harder de bladluizen lopen, hoe lager het percentage infectieuze bladluizen wordt. Het verschil in percentage infectieuze bladluizen in simulaties waar hard gelopen wordt (175 cm/uur) en waar zacht gelopen wordt (75 cm/uur) is vrij klein (ruim 8%) en misschien niet significant. Een verklaring hiervoor zou kunnen zijn, dat bij hoge loopsnelheden de

verspreidingssnelheid zo hoog wordt, dat veel bladluizen al weg zijn van de haard met besmette planten voordat ze het virus uit de planten op hebben kunnen nemen. Er ontstaat daardoor een groot gebied waarin zich bladluizen bevinden en een relatief klein gebied in het midden hiervan met besmette planten.

Indien er al verschil is in de maximum afstand die een bladluis in veldjes met vroeg- en in veldjes met laatgezaaide bieten per uur kan afleggen, dan is het waarschijnlijk dat deze afstand in het jonge gewas het grootst is. De LAI (Leaf Area Index) van het oude gewas is groter dan die van het jonge gewas. Dit betekent dat de bladluizen per m^2 grondoppervlak meer m^2 bladoppervlak hebben om op te lopen. De bietenplant wordt naarmate hij ouder wordt steeds meer een driedimensionale structuur waarin de bladluis niet alleen in een horizontale richting maar ook in een verticale richting kan lopen. Alle afstand die een bladluis in de verticale richting loopt, gaat ten koste van zijn maximum loopsnelheid in het horizontale vlak. Dit zou een grotere efficiëntie van bladluizen in laatgezaaide bieten betekenen en dus een hogere mate van vergeling.

-5- De predatiedruk

De predatiedruk heeft weliswaar invloed op het percentage infectieuze bladluizen in de simulaties, maar geen gerichte invloed. Bovendien zijn de verschillen klein (45 +/- 5%). Waarschijnlijk zijn er twee elkaar tegenwerkende krachten, die beide het percentage infectieuze bladluizen beïnvloeden. Door een hoge predatiedruk neemt de gemiddelde leeftijd van de bladluizen af en daarmee de verspreidingssnelheid. Bij

een lage verspreidingssnelheid hoort een hoog percentage infectieuze bladluizen. Door de hoge predatiedruk echter is de kans dat een bladluis het virus uit een plant op kan nemen en ook weer aan een andere plant over kan dragen klein. De kans om voordien gepredeerd te worden is groot.
Indien in veldjes met vroeggezaaide bieten de predatiedruk hoger is dan in veldjes met laatgezaaide bieten, kan dit mede verklarend zijn voor het verschil in vergeling. Bij een hoge predatiedruk neemt immers de efficiëntie af waarmee de bladluizen het virus over kunnen dragen. Dit verschil is in de veldproeven echter niet gevonden (zie tabel 1a en 1b).

-6- AFP en TFP

Hoe hoger de AFP en TFP, hoe hoger het percentage infectieuze bladluizen tijdens een simulatie. Van de AFP wordt een effect op het percentage infectieuze bladluizen verwacht. Zou de AFP erg hoog zijn, dan zou er geen virus meer opgenomen worden. Het percentage infectieuze bladluizen zou dan 0 zijn.
Van de TFP wordt een effect op de efficiëntie van virusoverdracht verwacht. Zou de TFP erg hoog zijn, dan zouden er geen planten meer geïnfecteerd worden. De efficiëntie zou dan 0 zijn. In de gevoeligheidsanalyse zijn de AFP en de TFP gelijktijdig gevariëerd. Er is alleen een effect op het percentage infectieuze bladluizen gevonden. Blijkbaar is de TFP geen beperkende factor geweest in de simulaties.
Wat het effect van de ouderdom van de bieteplant is op de AFP en de TFP is niet bekend. Er zijn geen verschillen gevonden in de vatbaarheid van planten met verschillende leeftijden (Kempenaar, 1987). Aangezien oude planten wat

dikkere en stuggere bladeren hebben dan jonge planten lijkt het aannemelijk, dat de AFP en de TFP hoger worden naarmate de planten ouder zijn. Dit zou mede verklarend kunnen zijn voor een hogere mate van vergeling in veldjes met laatgezaaide bieten.

-7- Voorkeur van de bladluizen voor jonge bladeren

Hoe hoger de voorkeur van de bladluizen voor het jonge bietenblad, hoe hoger het percentage infectieuze bladluizen tijdens de simulaties is. Dit is niet verwonderlijk aangezien het virus als eerste uit de jonge hartblaadjes voor de bladluizen opneembaar wordt. Het lijkt niet onredelijk een voorkeur van de bladluizen voor jonge bladeren te veronderstellen. De jonge hartblaadjes vormen de beste voedselbron voor de bladluizen. Omdat de jonge blaadjes van jonge planten aantrekkelijker zijn dan de jonge blaadjes van oude planten (Williams, 1987), zou een voorkeur voor jonge bladeren de vergeling in veldjes met laatgezaaide bieten sterker kunnen bevorderen dan in veldjes met vroeggezaaide bieten.

-C- Factoren die waarschijnlijk het percentage infectieuze bladluizen in de simulatie kunnen beïnvloeden maar waar geen gevoeligheidsanalyse voor gedaan is en die in het model als constant verondersteld zijn.

-8- De retentie periode.

Hoe langer de retentie periode is, hoe hoger het percentage infectieuze bladluizen in de simulaties waarschijnlijk zal zijn. De kans dat een bladluis lang genoeg gezogen heeft op

een geïnfecteerde plant om het virus op te nemen wordt immers steeds groter naarmate de lengte van de latentie-periode toeneemt.

-9- Voorkeur van bladluizen voor vergeelde bladeren.
Hier geldt het zelfde als bij de voorkeur van bladluizen voor jonge bladeren. Ook uit vergeelde bladeren kan altijd het virus opgenomen worden door de bladluizen. Uit proeven (Williams, 1987) is gebleken dat de groei en de reproductie van bladluizen op door virus vergeeld blad hoog zijn, hoger zelfs dan op jonge bladeren van jonge planten. Het is derhalve niet onredelijk om te veronderstellen dat bladluizen een voorkeur voor vergeeld bietenblad zouden hebben.

Hoofdstuk 6. Conclusies

De verspreiding van BYV in het veld hangt af van de efficiëntie waarmee de bladluizen het virus over kunnen dragen en van het aantal infectieuze bladluisdagen. De efficiëntie hangt samen met de verspreidingssnelheid van de bladluizen en het aantal infectieuze bladluisdagen. In het model is de verspreidingssnelheid van de bladluizen in vroeg- en in laatgezaaide bieten gelijk. In de simulaties is er een vast verband tussen het aantal infectieuze bladluisdagen en het aantal vergeelde planten.

Bovenstaande betekent voor de simulaties, dat het verschil in verspreiding van BYV in vroeg- en in laatgezaaide bieten volledig toegeschreven moet worden aan het verschil in aantal infectieuze bladluisdagen.

De hieronder genoemde factoren hebben invloed gehad op het percentage infectieuze bladluizen in de simulaties met de standaard opstartwaarden.

- Latentieperiode
- Percentage besmet bladoppervlak
- Percentage lopende bladluizen

Het percentage infectieuze bladluizen en het aantal bladluizen zijn in de veldjes met laatgezaaide bieten hoger. Hierdoor ligt het aantal infectieuze bladluisdagen in deze veldjes aanzienlijk hoger dan in veldjes met vroeggezaaide bieten.

Resultaten uit de gevoeligheidsanalyse

Er zijn tijdens de gevoeligheidsanalyse een aantal factoren naar voren gekomen die invloed hebben op de snelheid waarmee bladluizen in de simulatie planten kunnen infecteren (efficiëntie). Een hogere verspreidingssnelheid van de bladluizen verandert het verband tussen het aantal vergeelde planten en het aantal infectieuze bladluisdagen. Hoe hoger deze snelheid is, hoe meer planten er door eenzelfde aantal infectieuze bladluizen geïnfecteerd kunnen worden. De efficiëntie is dan dus hoger.

Van de hieronder genoemde factoren kan gezegd worden, dat de invloed die zij hebben op de snelheid waarmee een bladluis planten kan infecteren groter is, dan door hun invloed op het aantal infectieuze bladluisdagen in de simulatie verondersteld mag worden. Een invloed op de efficiëntie hebben derhalve.

- Predatie
- Maximum loopsnelheid
- Percentage lopende bladluizen

De AFP en de TFP, de latentieperiode en de voorkeur van bladluizen voor jonge bladeren beïnvloeden het percentage infectieuze bladluizen en daarmee het aantal infectieuze bladluisdagen in de simulatie. De invloed op de efficiëntie is klein en moet waarschijnlijk aan toevalseffecten toegeschreven worden.

Tabel 3. De waarschijnlijke samenhang van de verschillende factoren met de mate van vergeling in het veld.

Het aantal vergeelde planten hangt af van:

- het aantal infectieuze bladluisdagen
  hangt af van:
  - het aantal bladluizen
    hangt af van:
    - de leeftijd van de bieteplanten
    - de predatiedruk
    - de temperatuur
  - het percentage infectieuze bladluizen
    hangt af van:
    - de AFP
    - de latentieperiode
    - het percentage besmet bladoppervlak
    - de voorkeur voor jonge/vergeelde bladeren
    - het percentage lopende bladluizen per uur
    - de retentieperiode
    - de predatiedruk
- de efficiëntie van virusoverdracht
  hangt af van:
  - de TFP
  - de verspreidingssnelheid van de bladluizen
    hangt af van:
    - de maximum loopsnelheid
    - het percentage lopende bladluizen per uur
    - de predatiedruk
    - de temperatuur

Hoewel dit model geen antwoord geeft op de vraag waar het loopgedrag van de bladluizen door beïnvloed wordt, geeft het wel inzicht in hoe de verspreidingssnelheid van BYV in het veld met het loopgedrag en met diverse andere factoren samen hangt. Bepalend voor de verspreidingssnelheid van het virus in het veld zijn: het aantal infectieuze bladluisdagen en de efficiëntie van virusoverdracht.

Aanbevelingen

- Tellingen van bladluisaantallen

De bladluisaantallen, die gebruikt zijn voor het bepalen van het aantal bladluizen in de simulaties, zijn geteld op negen "telplanten". De getelde waarden zijn het resultaat van reproductie, natuurlijke sterfte, predatie en verspreiding van de bladluizen en niet als laatste van de telcapaciteiten van de teller zelf. De invloeden van al deze factoren op het aantal getelde bladluizen is moeilijk te achterhalen.

De getelde aantallen bladluizen kunnen alleen van nut zijn indien de verspreidingssnelheid van de bladluizen en hun leeftijdsverdeling bekend is, omdat dan de predatiedruk en het totale aantal bladluizen in het veld achterhaald kunnen worden.

De leeftijdsverdeling van de getelde bladluizen bepalen is eenvoudig mogelijk. De verspreidingssnelheid van de bladluizen in het veld bepalen is veel ingewikkelder. Een eerste vereiste om de verspreiding van het virus in het veld beter te kunnen simuleren is, dat de verspreidingssnelheid van de bladluizen beter bekend wordt. Toekomstig onderzoek aan de vergeling in bietenveldjes zou hierop gericht moeten zijn.

- Verspreidingssnelheid van de bladluis als functie van het ontwikkelingsstadium van de bieteplant

Voor de kwaliteit van het model is het van groot belang een goede benadering van de verspreidingssnelheid van de bladluizen in het veld te hebben. Omdat de verspreidingssnelheid

waarschijnlijk afhankelijk is van het ontwikkelingsstadium van de bieteplant, zou de verspreidingssnelheid van de bladluizen in veldjes met bieten van verschilende ouderdom bepaald moeten worden. In het model is de verspreidingssnelheid van de bladluizen het resultaat van de loopsnelheid en het loopgedrag. De invloed van het ontwikkelingsstadium van de bieteplant zou voor beide afzonderlijk bepaald moeten worden.

- Percentage infectieuze bladluizen

In de simulaties is telkens het percentage infectieuze bladluizen bepaald. Voor het controleren van de juistheid van deze percentages zou het werkelijke percentage infectieuze bladluizen in het veld bepaald moeten worden. Dit is van belang, omdat de invloed van het aantal infectieuze bladluizen in het veld op de uiteindelijke mate van vergeling in het veld groot is.

- AFP en TFP

Vooral de AFP heeft een grote invloed op het aantal infectieuze bladluizen in de simulatie. Het is daarom van belang uit te zoeken hoe lang deze periode precies is en of de lengte van de periode afhangt van het ontwikkelingsstadium van de bieteplant.

Overige punten niet in het model verwerkt, maar mogelijkerwijs wel van invloed op de snelheid van verspreiding van bladluizen en/of van het virus in het veld.

- Afwijkende groei bij besmette bieteplanten.
- Angstferomoon bij predatie van bladluizen (loopmotivatie).

Literatuur

Baker,P.F. 1960. Aphid behaviour on healthy and on yellows virus infected sugar beet. Ann. appl. Biol. 48: 384-391.

Clark,E.A. and R.S.Loomis. 1978. Dynamics of leaf-growth and development in sugarbeet. J.of A.S.S.B.T.20 (1978): vol 2.

Duffus,J.E. 1973. The yellowing virus diseases of beet. Adv.Virus Res. 18 (1973): 347-386.

Ferrar,P. 1968. Interplant movement of apterous aphids with special reference to Myzus persicae (Sulz.) (Hem., Aphididae).

Heathcote,G.D. 1964. Transmission of beet yellows virus by alate and apterous aphids. i.appl.Biol. 53:259-266.

Heathcote,G.D. 1970. Effects of plant spacing and time of sowing of sugarbeet on aphid infestation and spread of virus yellows. Pl. Path. 19 (1970): 32-39.

Heijbroek,W. 1984. De vergelingsziekte waarschuwingsdienst, 25 jaar geïntegreerde bestrijding. Instituut voor Rationele Suikerproductie, Bergen op Zoom.

Hille Ris Lambers,D. 1972. Aphids: their life cycles and their role as virus vectors. In: Viruses of potatoes and seed potato production. Ed. J.A. de Bokx. Pudoc, Wageningen.

Jansson,R.K. and Z.Smilowitz. February 1985. Development and reproduction of the green peach aphid, Myzus persicae (Homoptera: Aphididae), on upper and lower leaves of three potato cultivars. Can.Ent. 117:247-252.

Jepson,P.C. 1983. A controlled environment study of the effect of leaf physiological age on the movement of apterous Myzus persicae on sugar-beet plants. Ann. appl. Biol. 103: 173-183.

Kempenaar,C. 1987. Systeemanalyse van de verspreiding van BYV in vroeg en laat gezaaide suikerbieten. Doctoraalverslag TPE, LUW.

Milford,G.F.J., T.O.Pocock and J.Riley. 1985. An analysis of leaf growth in sugar beet. I. Leaf appearance and expansion in relation to temperature under controlled conditions. Ann. appl. Biol. 106:163-172.

Milford,G.F.J., T.O.Pocock and J.Riley. 1985. An analysis of leaf growth in sugar beet. II. Leaf appearance in field crops. Ann. appl. Biol. 106:173-185.

Milford,G.F.J., T.O.Pocock, J.Riley and A.B.Messen. 1985. An analysis of leaf growth in sugar beet. III. Leaf expansion in field crops. Ann. appl. Biol. 106:187-203.

Milford,G.F.J., T.O.Pocock, K.W.Jaggard, P.V.Biscoe, M.J.Armstrong, P.J.Last and P.J.Goodman. 1985. An analysis of leaf growth in sugar beet. IV. The expansion of the leaf canopy in relation to temperature and nitrogen. Ann. appl. Biol. 107:335-347.

Russel,G.E. 1965. The host range of some english isolates of beet yellowing viruses. Ann. Apl. Biol. 55 (1965): 245-252.

Sabelis,M.W. 1981. Biological control of two-spotted spider mites using phytoseiid predators. Part I (LHW).

Smith,K.M. 1972. A textbook of plant virus diseases. Third edition: 91-110.

Sylvester,E.S. 1961. Re-evaluation of retention of the beet yellows virus by green peach aphids under fasting and feeding conditions. Virology. 14:476-479.

Watson,M.A. 1940. Studies on the transmission of sugar-beet yellows virus by the aphis, Myzus persicae (Sulz.). Proc.Roy.Soc.B. 125:535-552.

Watson,M.A. 1946. The transmission of beet mosaic and beet yellows viruses by aphides; a comparative study of a non-persistent and a persistent virus having host plants and vectors in common. Proc.Roy.Soc.B. 133:200-220.

Watson,M.A., G.D.Heathcote, F.B.Laukner and P.A.Sowray. 1975. Tje use of weather data and counts of aphids in the field to predict the incidence of yellowing viruses of sugarbeet crops in England in relation to the use of insecticides. Ann. appl. Biol.81:181-198.

Werf,W. van der. 1988. Yellowing viruses in sugarbeet; epidemiology and damage. Proefschrift (LUW).

Williams,C.T. 1987. Effects of plant growth stage and condition on the population dynamics of Myzus persicae Sulz. on sugar beet in field plots. Manuscript, Department of Applied Biology, University of Cambridge and Broom's Barn Experimental Station, Higham.

BIJLAGEN

Toelichting bij de resultaten van een simulatie met vroeg-gezaaide (18 april, bijlage A) en een simulatie met laat-gezaaide bieten (20 mei, bijlage B).

Op iedere pagina staat hetzelfde bietenveldje twee keer afgebeeld. De rijen liggen horizontaal. In het bovenste veldje zijn de besmettingsklassen en de vergeling van de bieteplanten uitgezet. In het andere veldje de bladluizen-aantallen per bieteplant. Er ligt een periode van een week tussen de opeenvolgende pagina's. Beide simulaties zijn gestart op 23 juni.

In het bovenste veldje representeert elk getal (0 t/m 9) en elk rastertje (▒, ▓, ▒ en ▒) een bieteplant. Het getal geeft de besmettingsklasse van de plant aan (vb. 30% < klasse4 <= 40%). Indien een bieteplant vergeeld is, wordt er een rastertje in plaats van een getal voor de plant afgebeeld. Hoe donkerder het rastertje, hoe hoger de besmettingsklasse van de plant (▒ klasse 0-3, ▓ klasse 3-6, ▒ klasse 6-9 en ▒ klasse 9-10).

In het onderste veldje staat elk getal (0 t/m 9) voor het aantal bladluizen dat zich op de betreffende bieteplant bevindt. Indien er meer dan 9 bladluizen op een plant zitten, wordt er een rastertje (▒) afgebeeld.

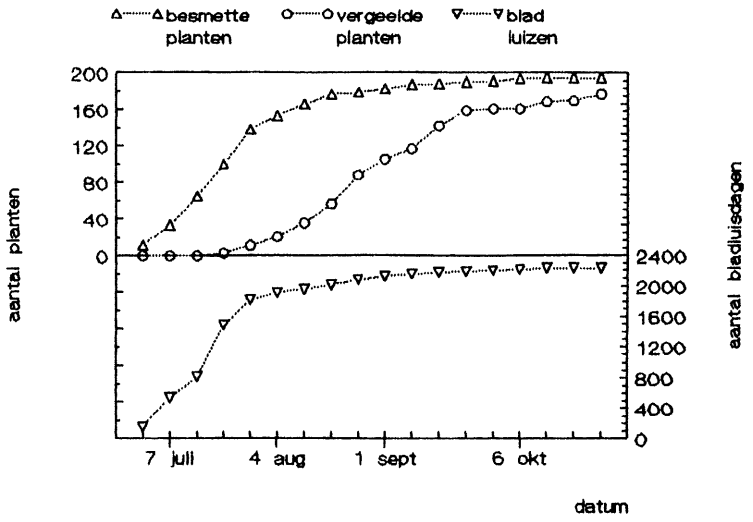
Enkele opvallende aspecten van de simulaties:

- In beide simulaties grijpt de epedemie al in volle sterkte om zich heen terwijl er in het veld nog geen symptomen waar te nemen zijn. Dit geldt vooral voor de simulatie met de vroeggezaaide bieten vanwege de lange incubatieperiode.
- De besmettingsklassen van de planten zijn in de simulatie met de laatgezaaide bieten duidelijk het hoogst.
- In de simulatie met de laatgezaaide bieten is de vergeling en het aantal bladluizen het grootst.
- De verspreiding van de bladluizen verschilt niet veel in beide simulaties.

Van beide simulaties is het aantal besmette planten, het aantal vergeelde planten en het aantal bladluizen uitgezet tegen de tijd. Figuur a is van de simulatie met de vroeggezaaide bieten (bijlage A). Figuur b is van de simulatie met de laatgezaaide bieten (bijlage B). In figuur c staan alle grafieken nog eens uitgezet maar nu met voor alle grafieken dezelfde y as schaal.

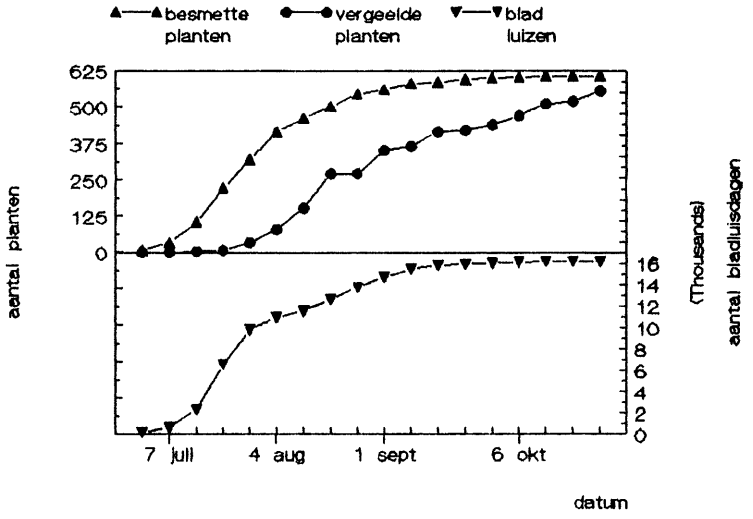
a

een simulatie met vroeg gezaaide bieten



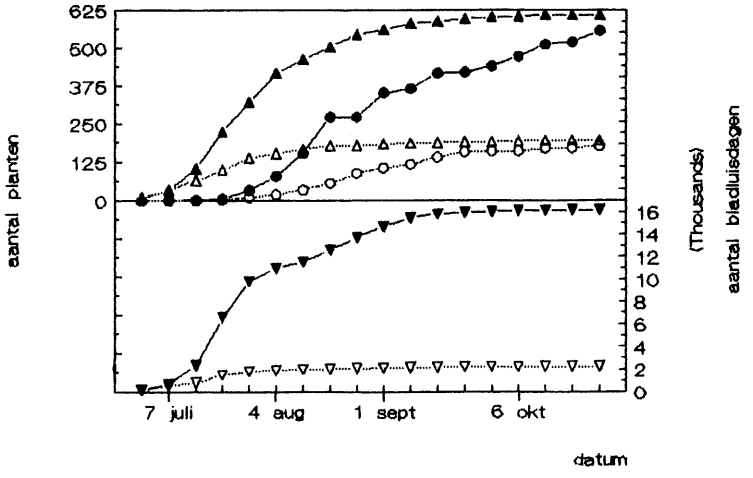
b

een simulatie met laat gezaaide bieten



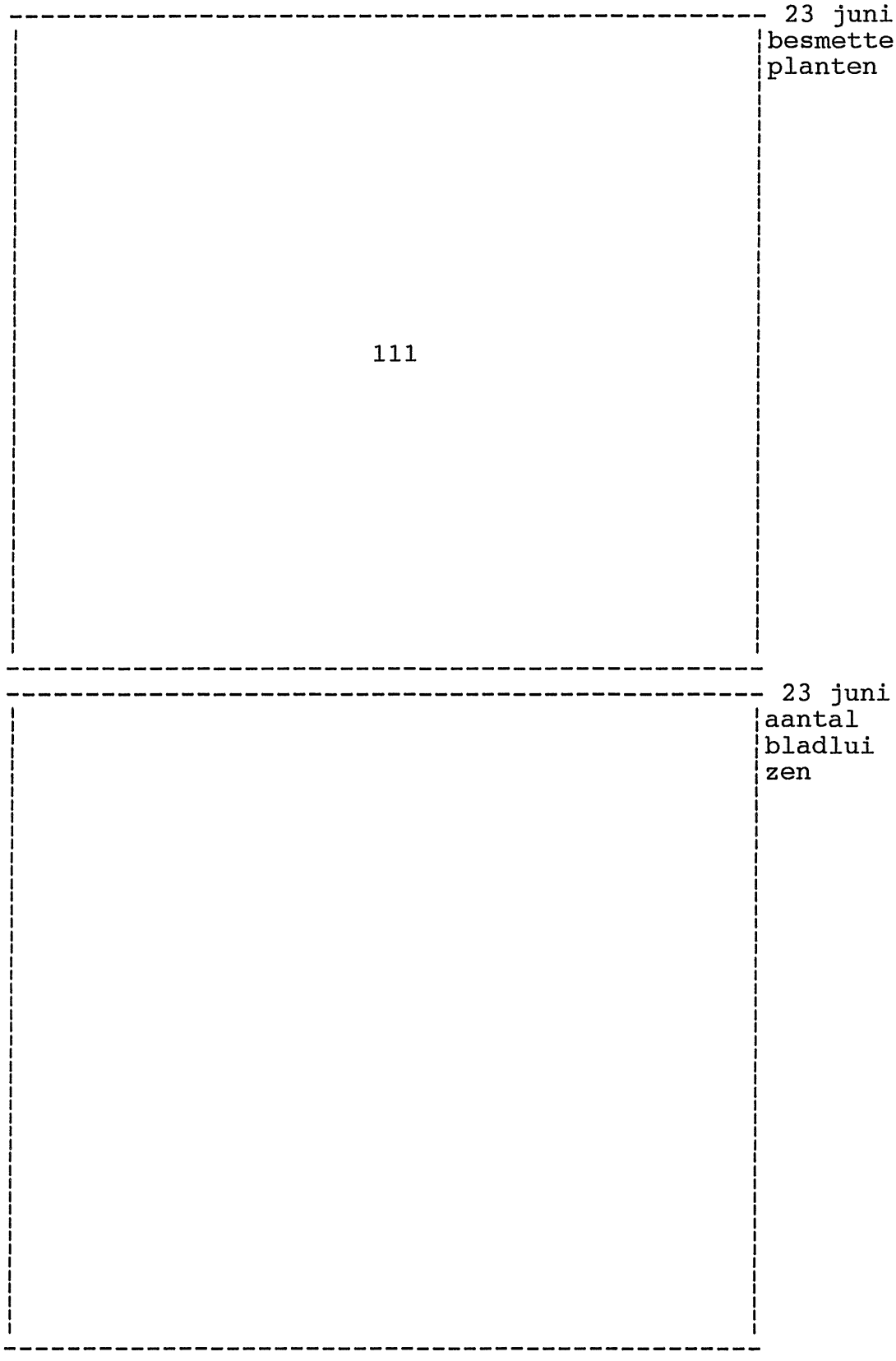
c

de resultaten van beide simulaties in één grafiek

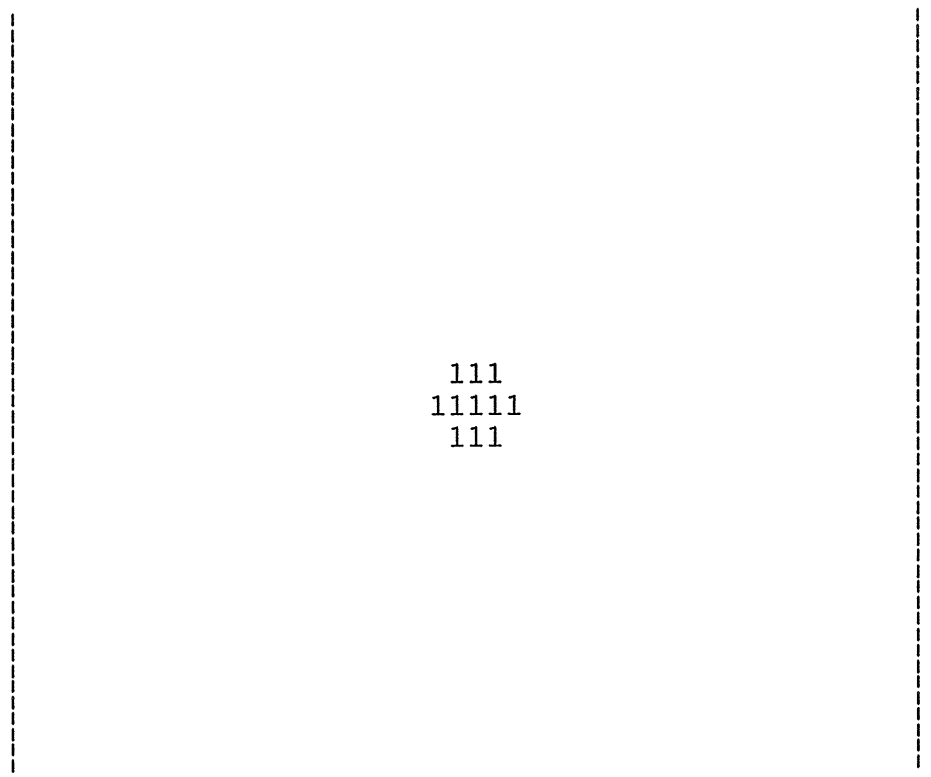


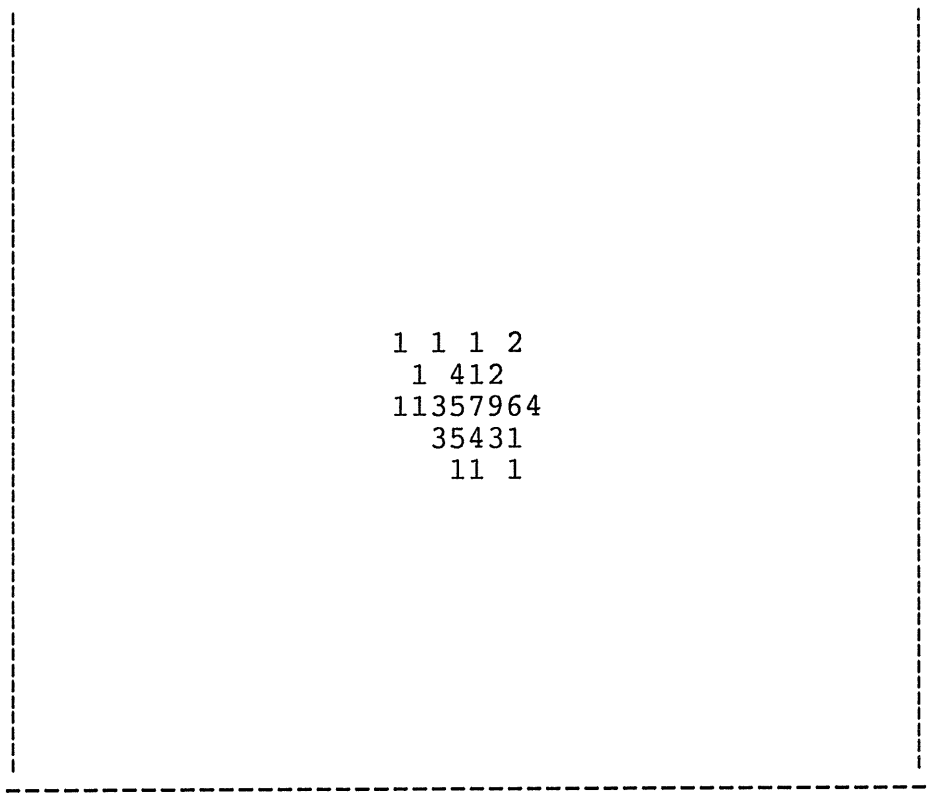
Bijlage A

Een simulatie met vroeggezaaide bieten (18 april 1986) met de luizentellingen uit file luizen2.dat.

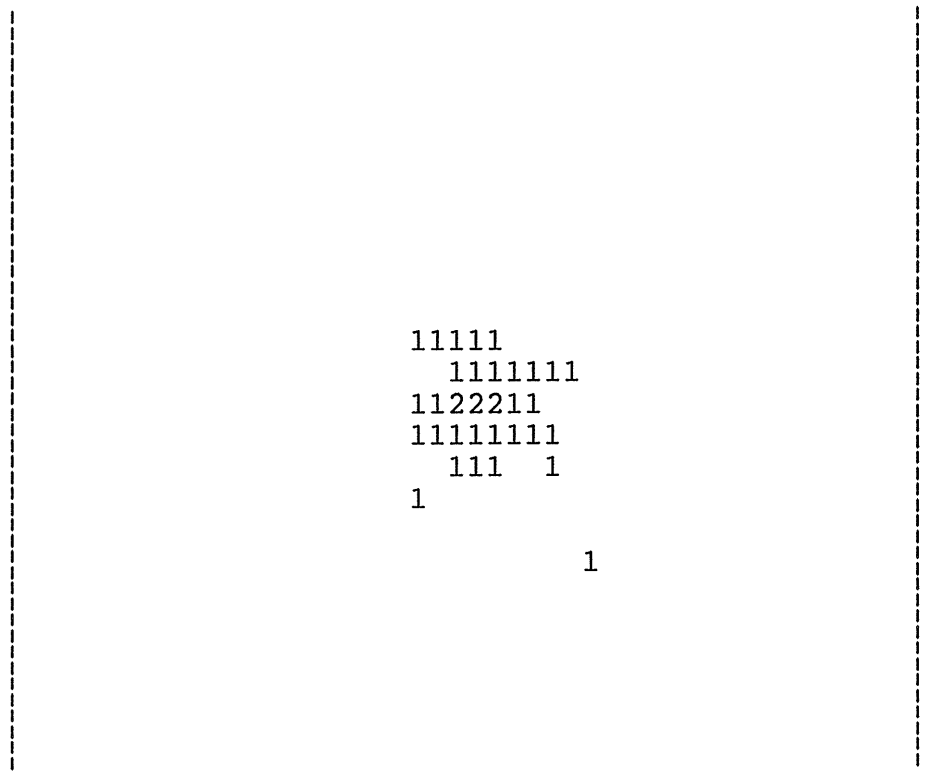


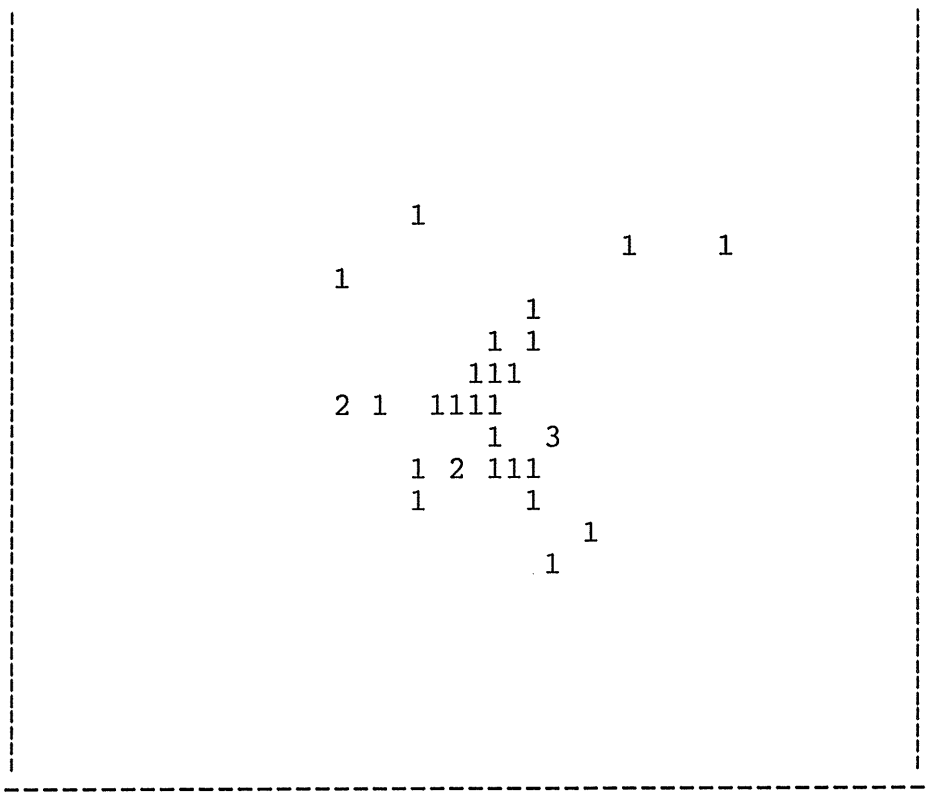
30 juni



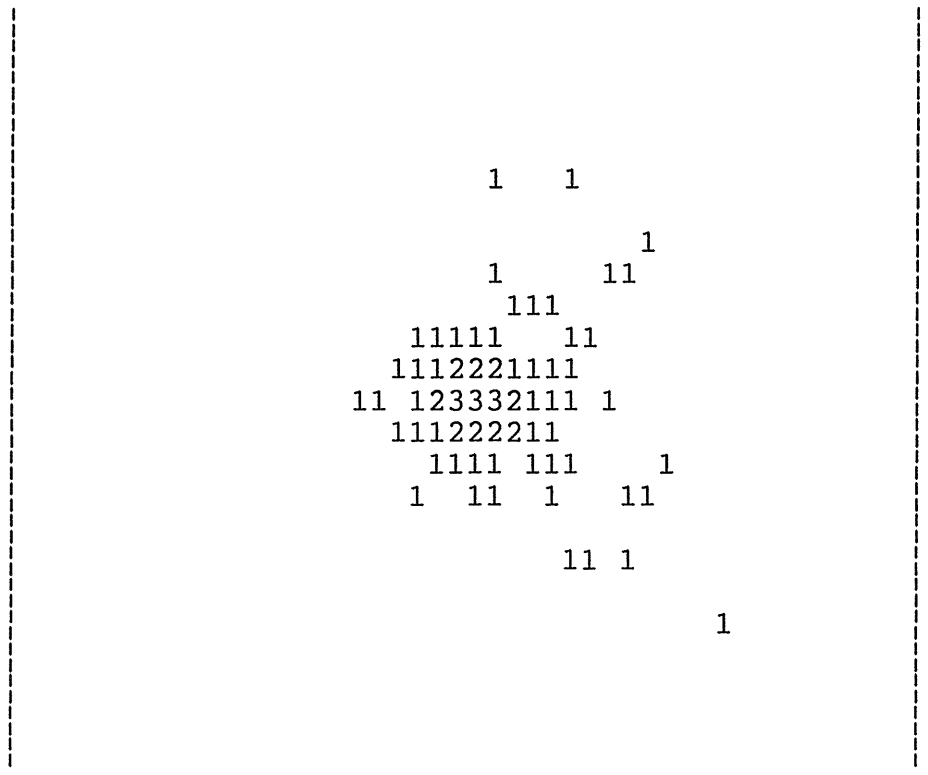


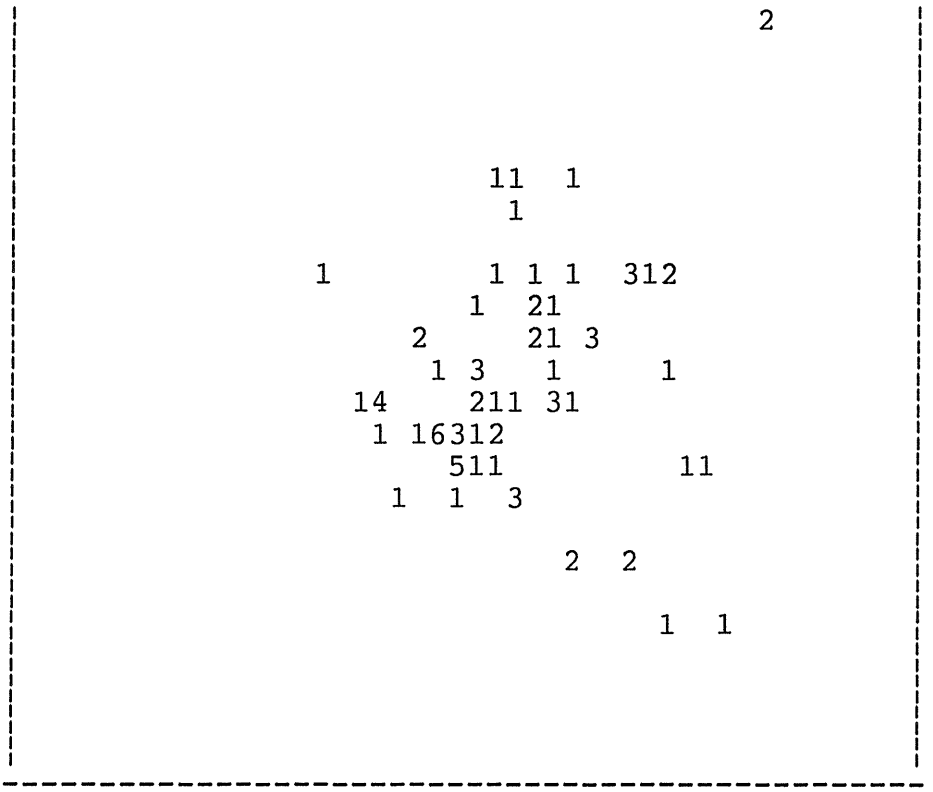
7 juli


11111
1111111
1122211
11111111
111 1
1
1

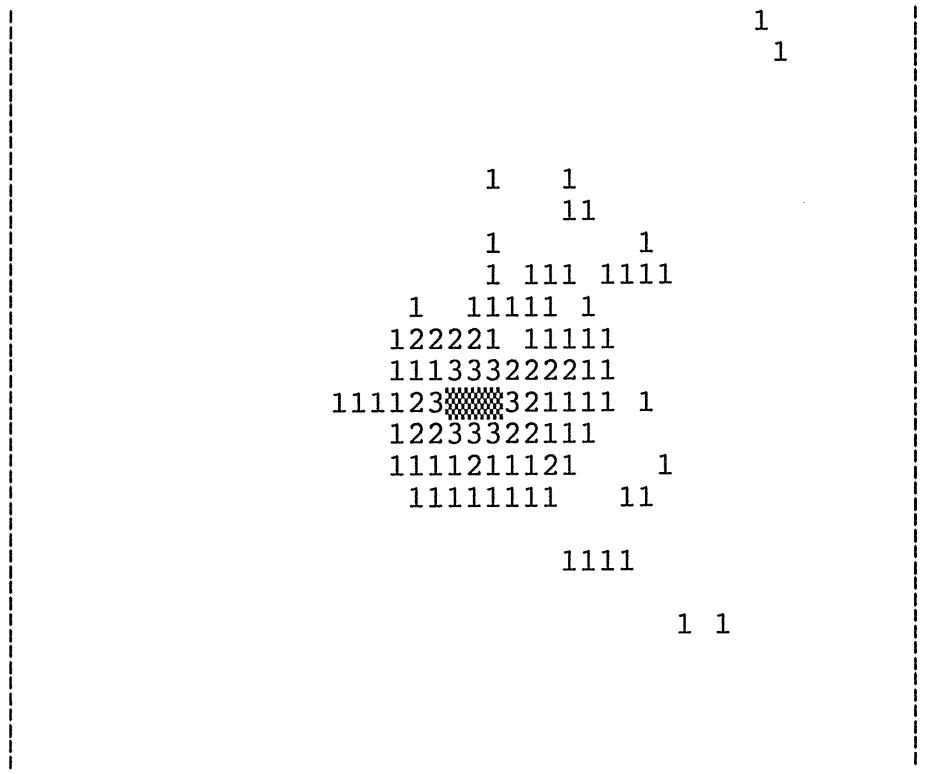

1
1 1
1
1
1 1
111
2 1 1111
1 3
1 2 111
1 1
1
1

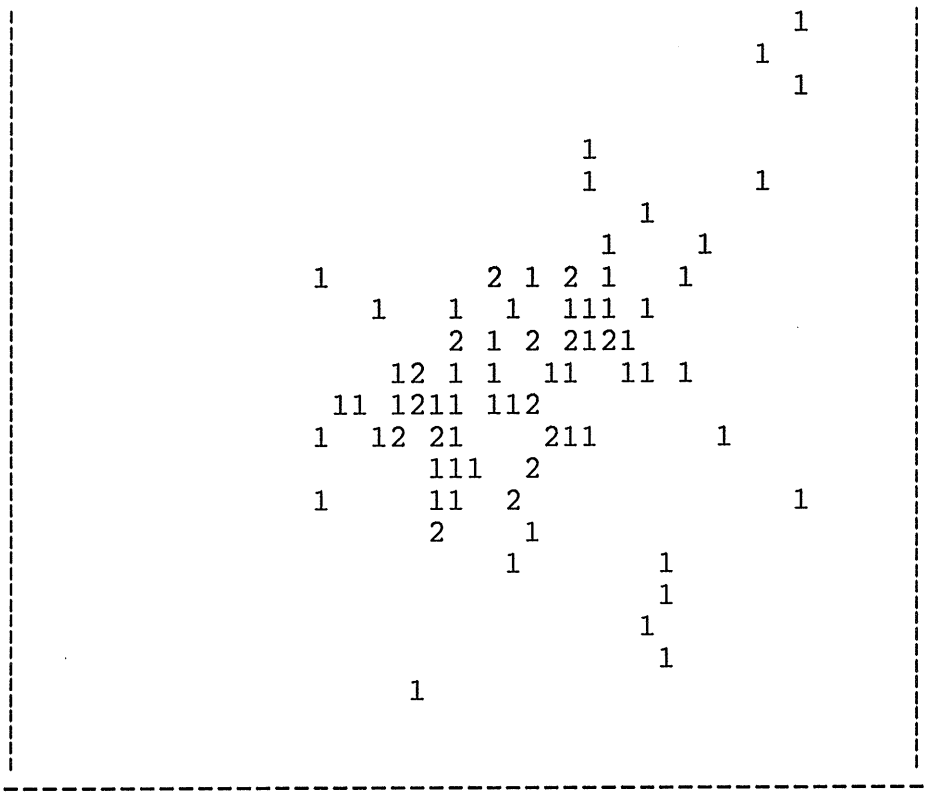
14 juli


1 1
1
1 11
111
11111 11
1112221111
11 123332111 1
111222211
1111 111 1
1 11 1 11
11 1
1

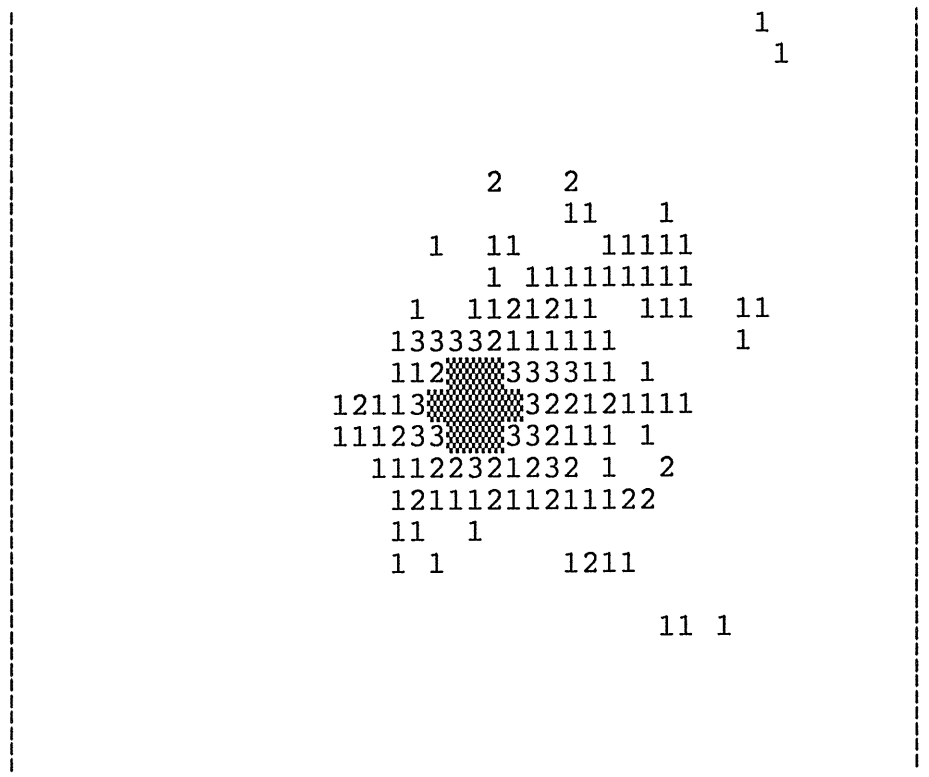

2
11 1
1
1 1 1 1 312
1 21
2 21 3
1 3 1 1
14 211 31
1 16312
511 11
1 1 3
2 2
1 1

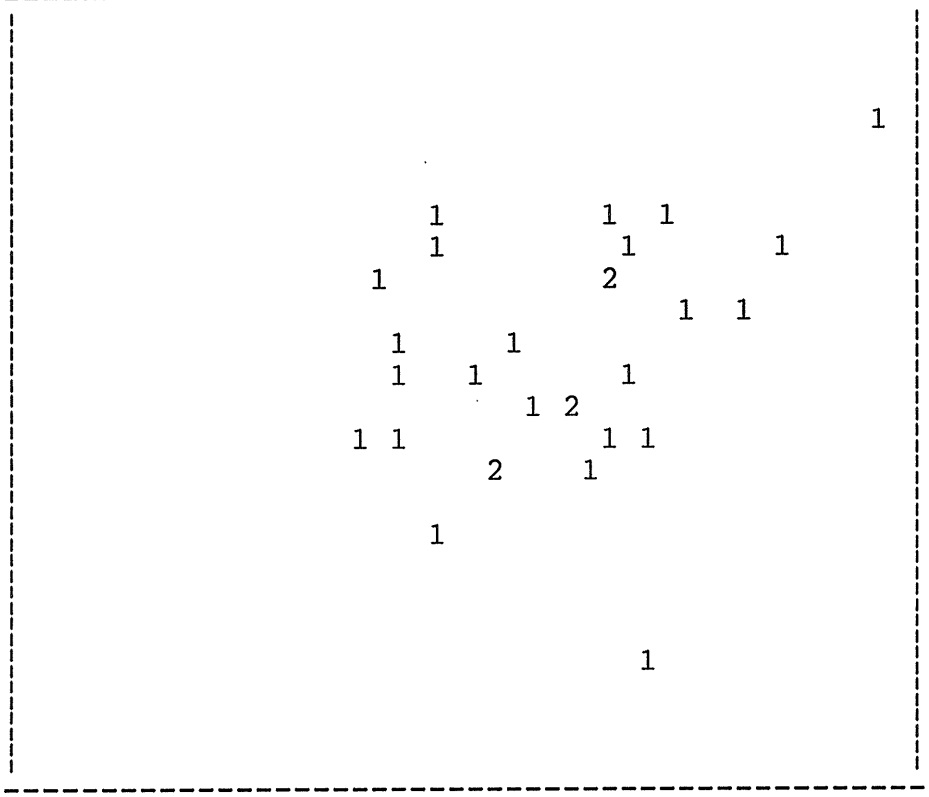
21 juli



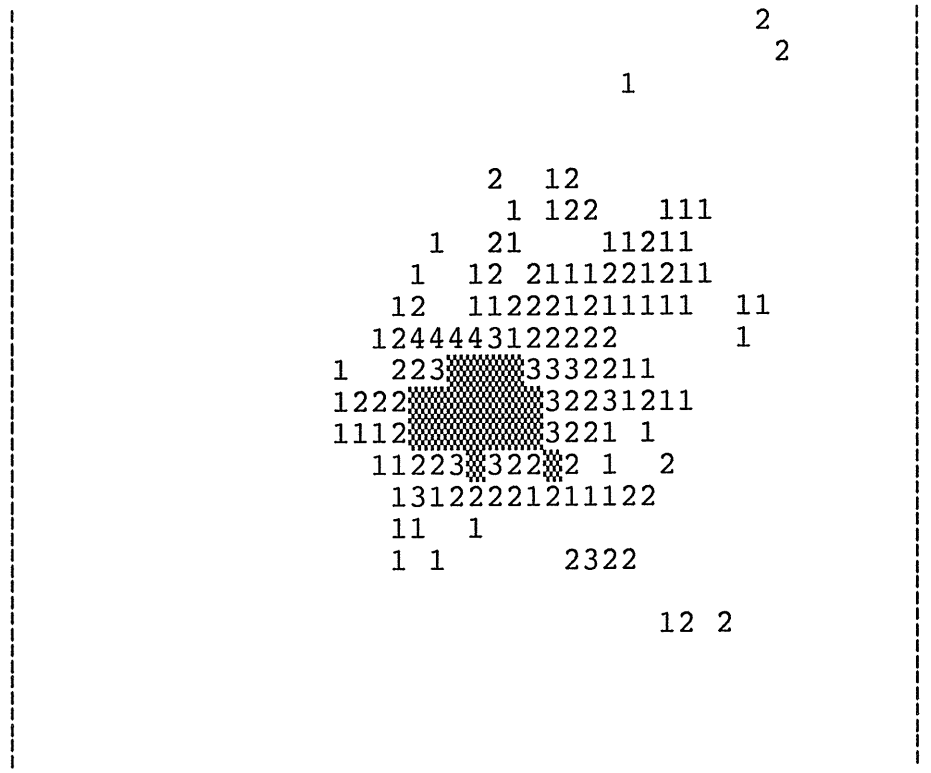


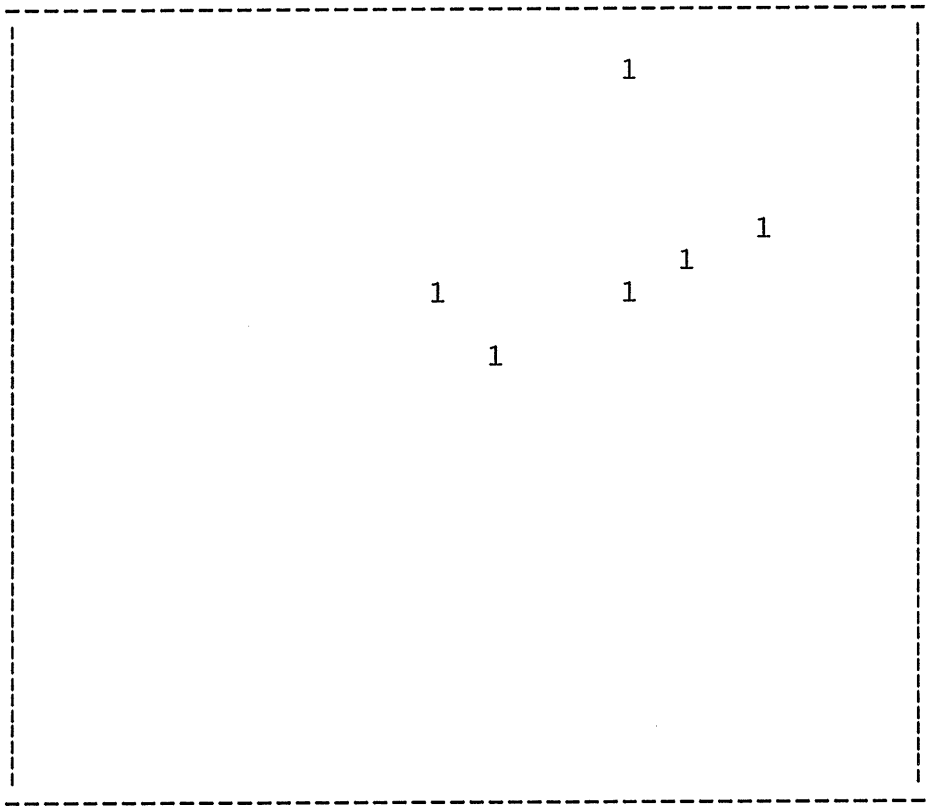
28 juli



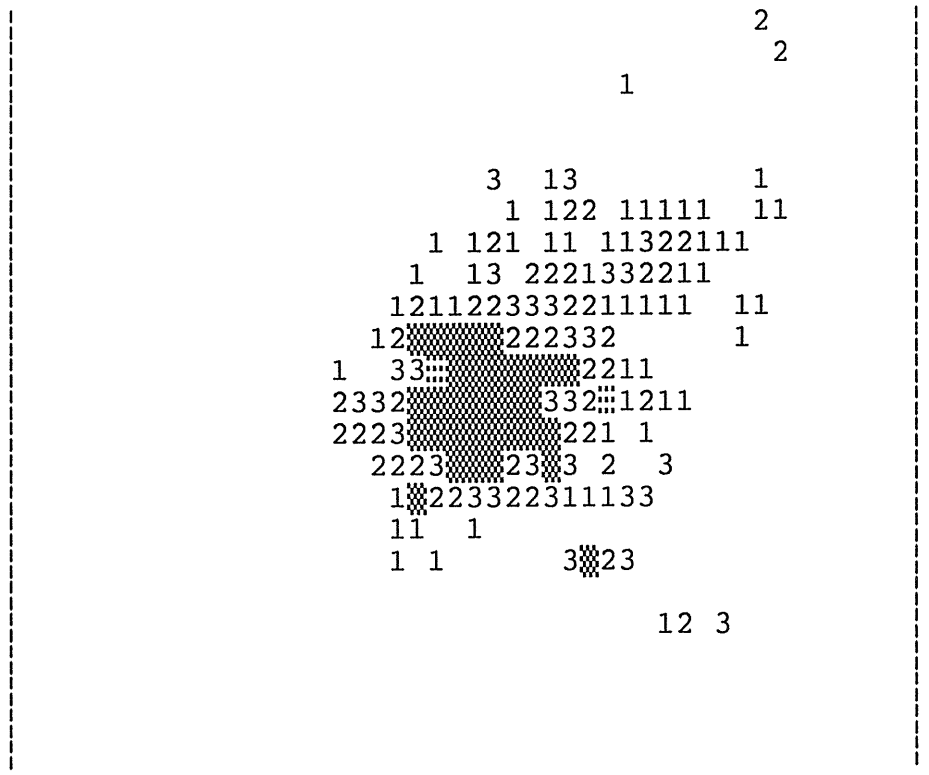


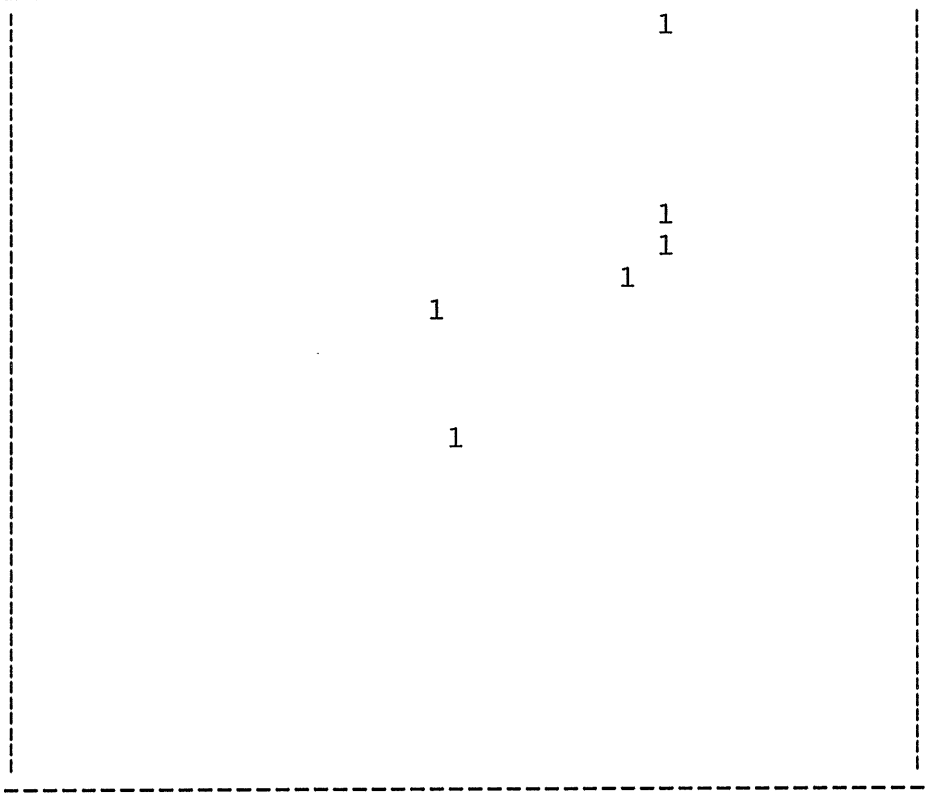
4 aug



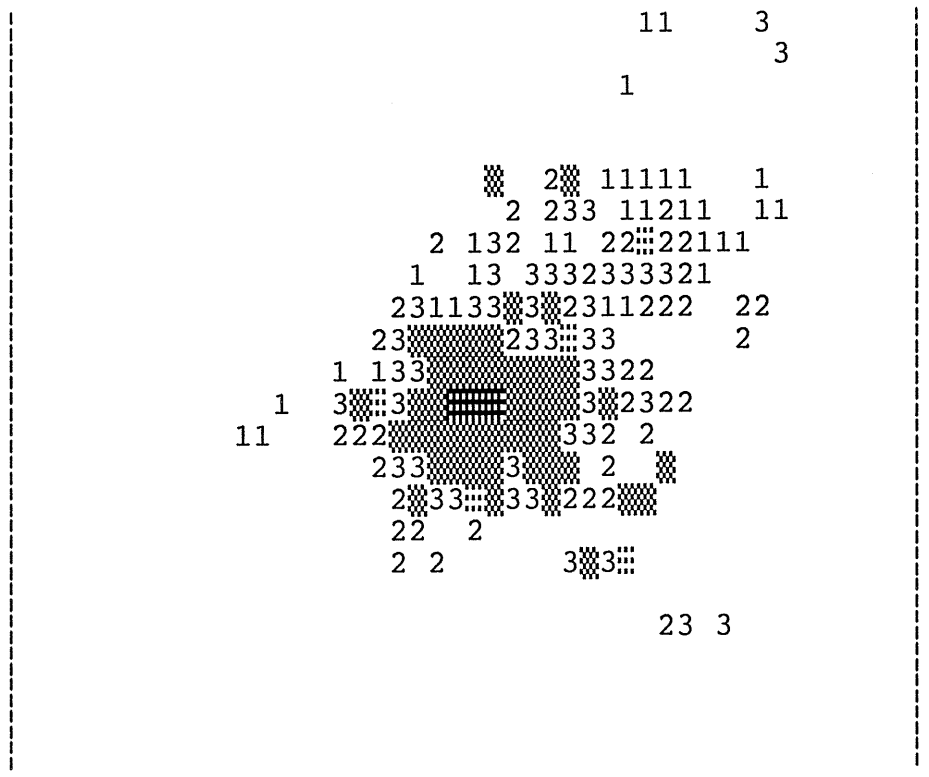


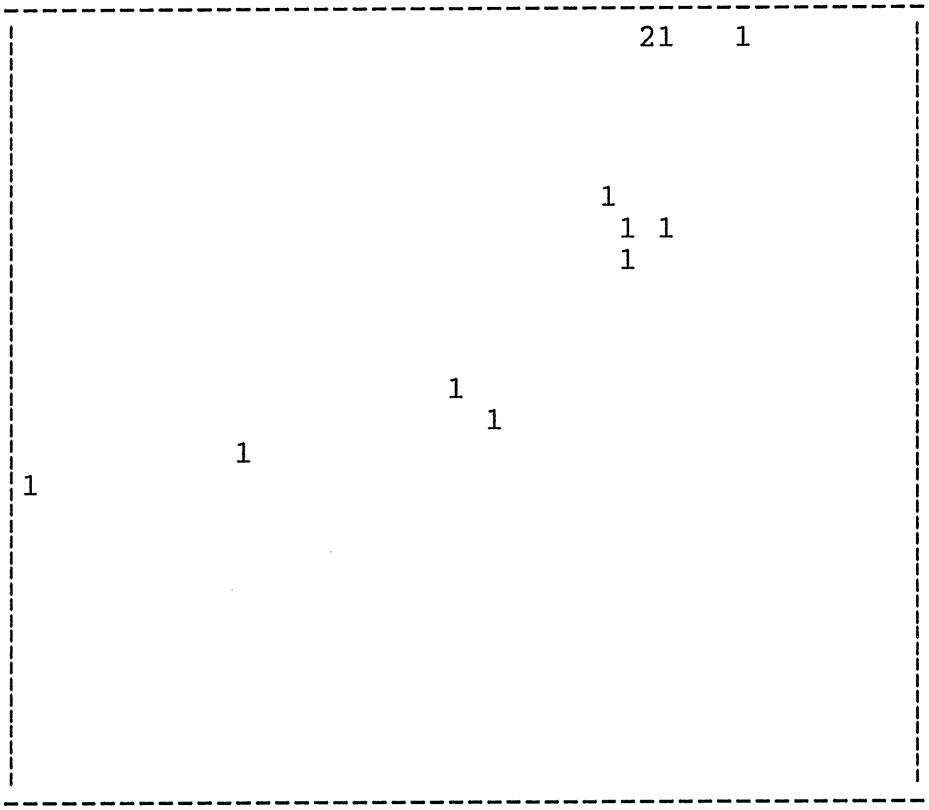
11 aug



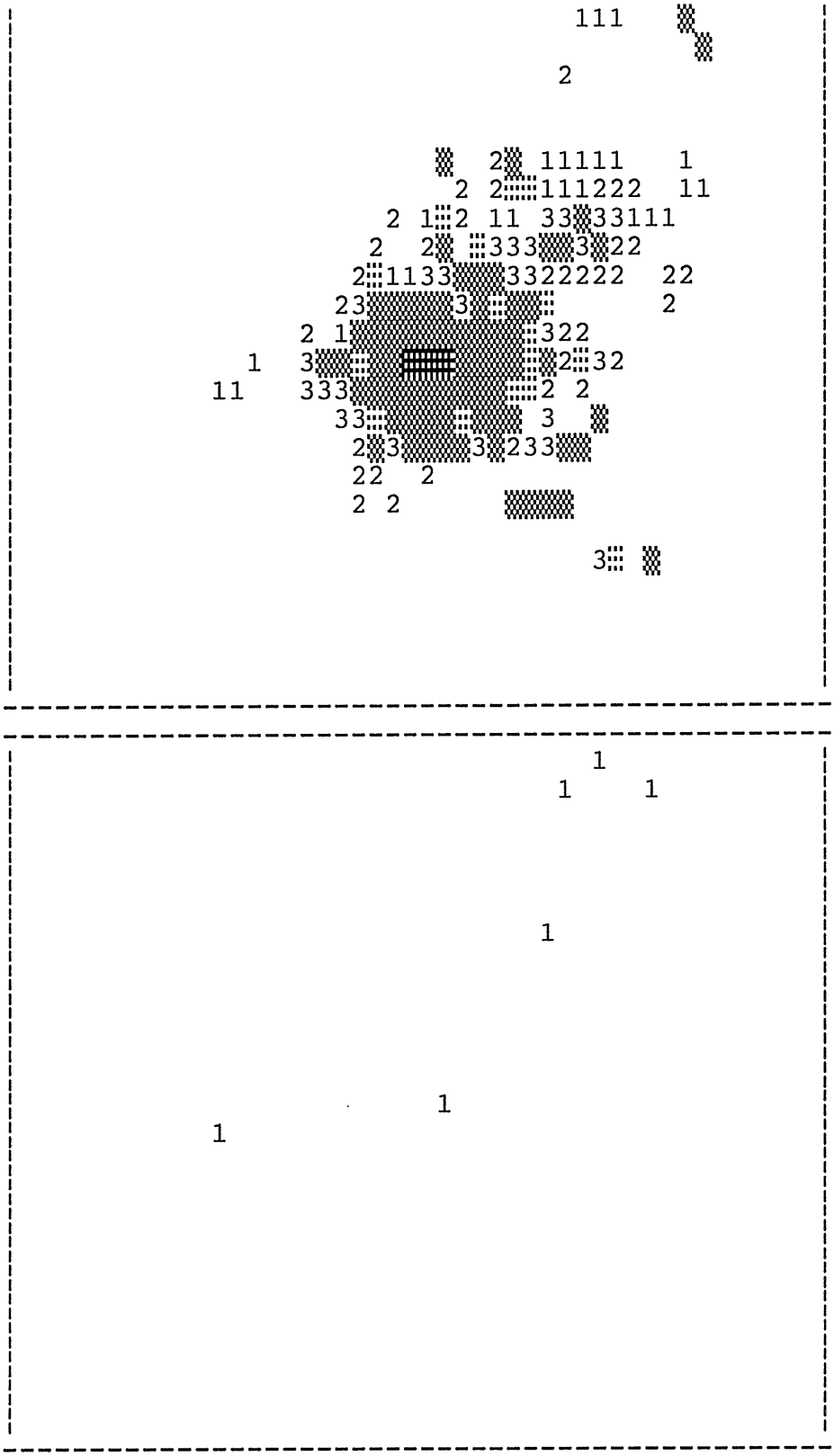


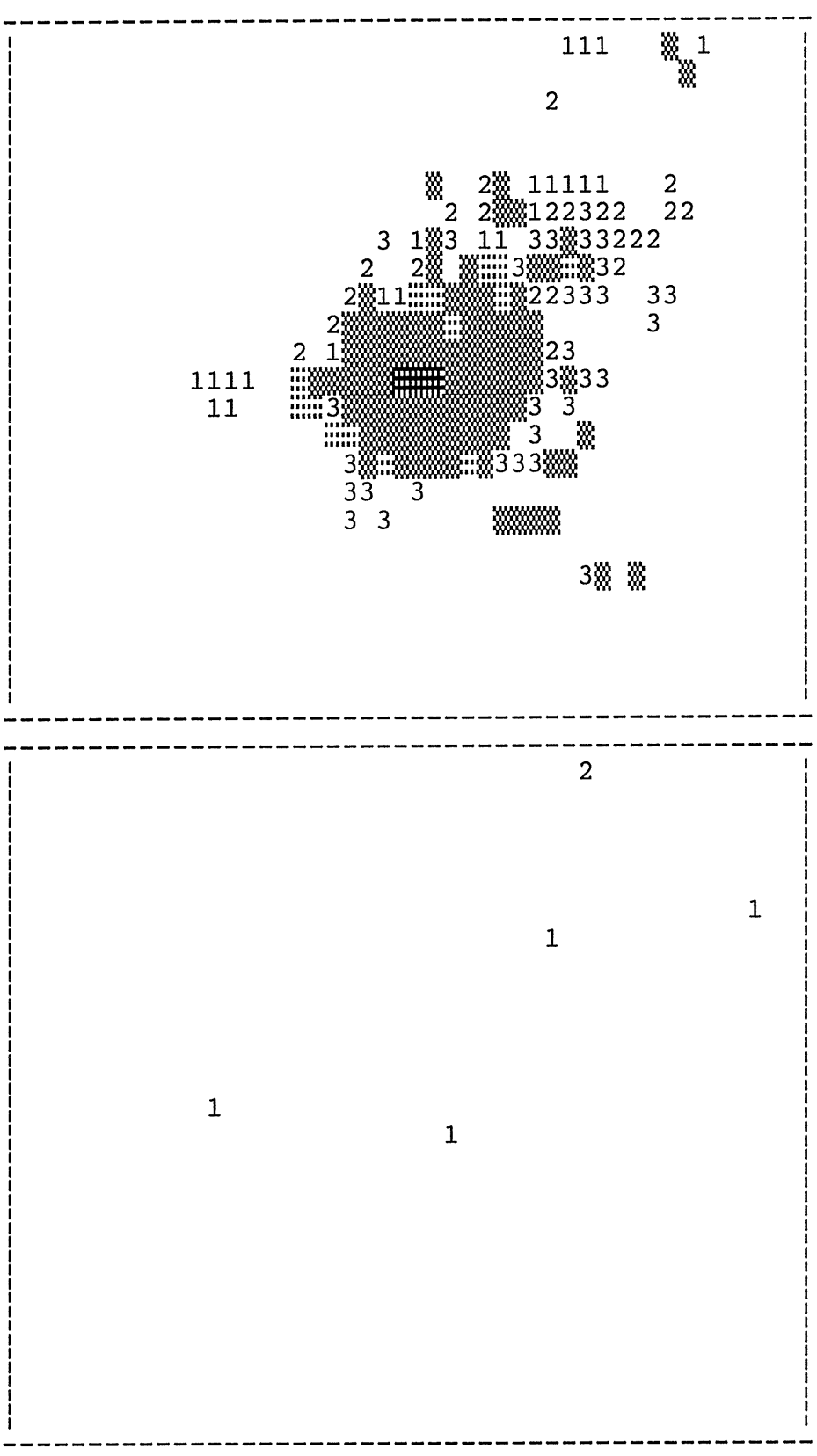
18 aug



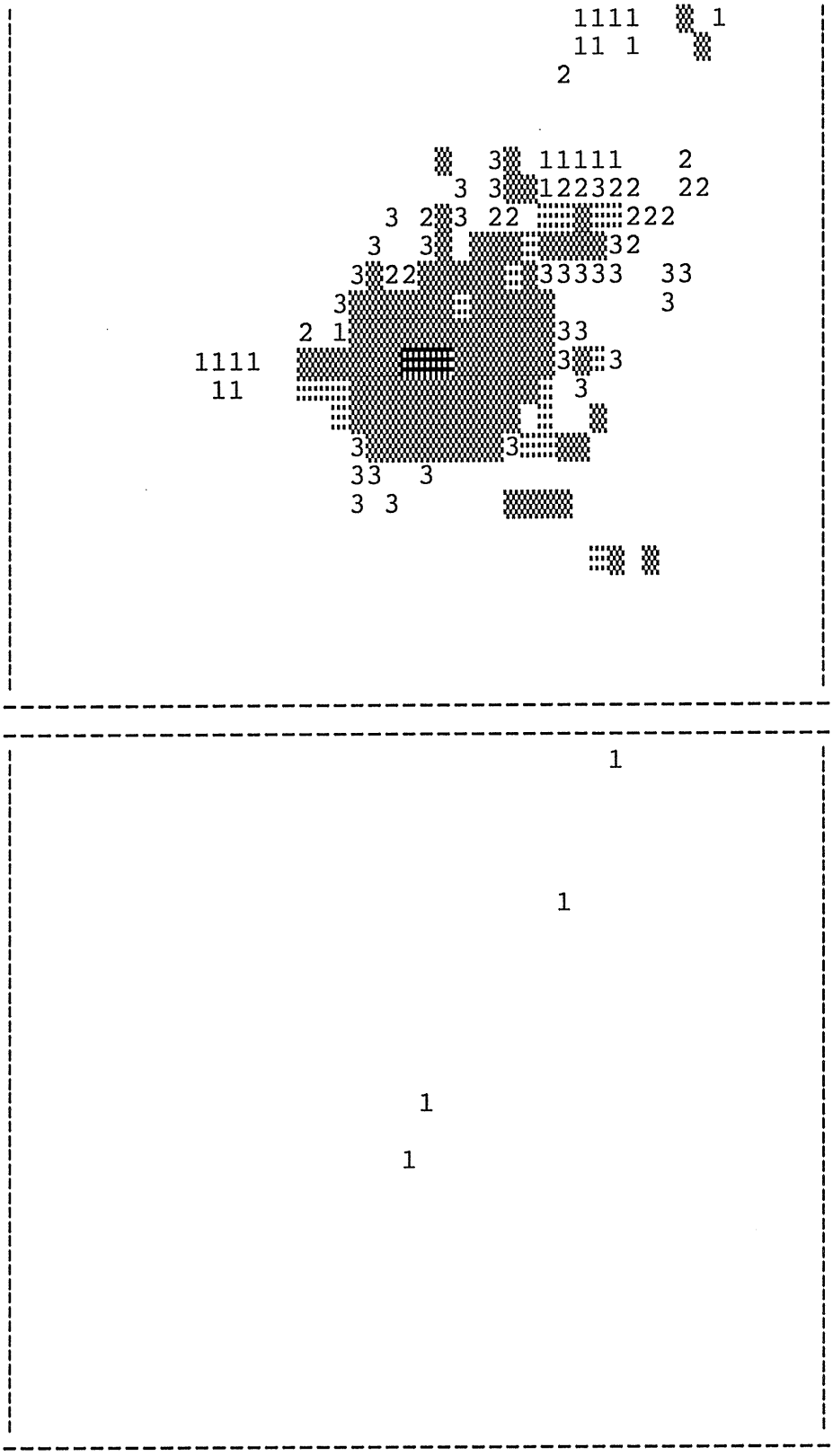


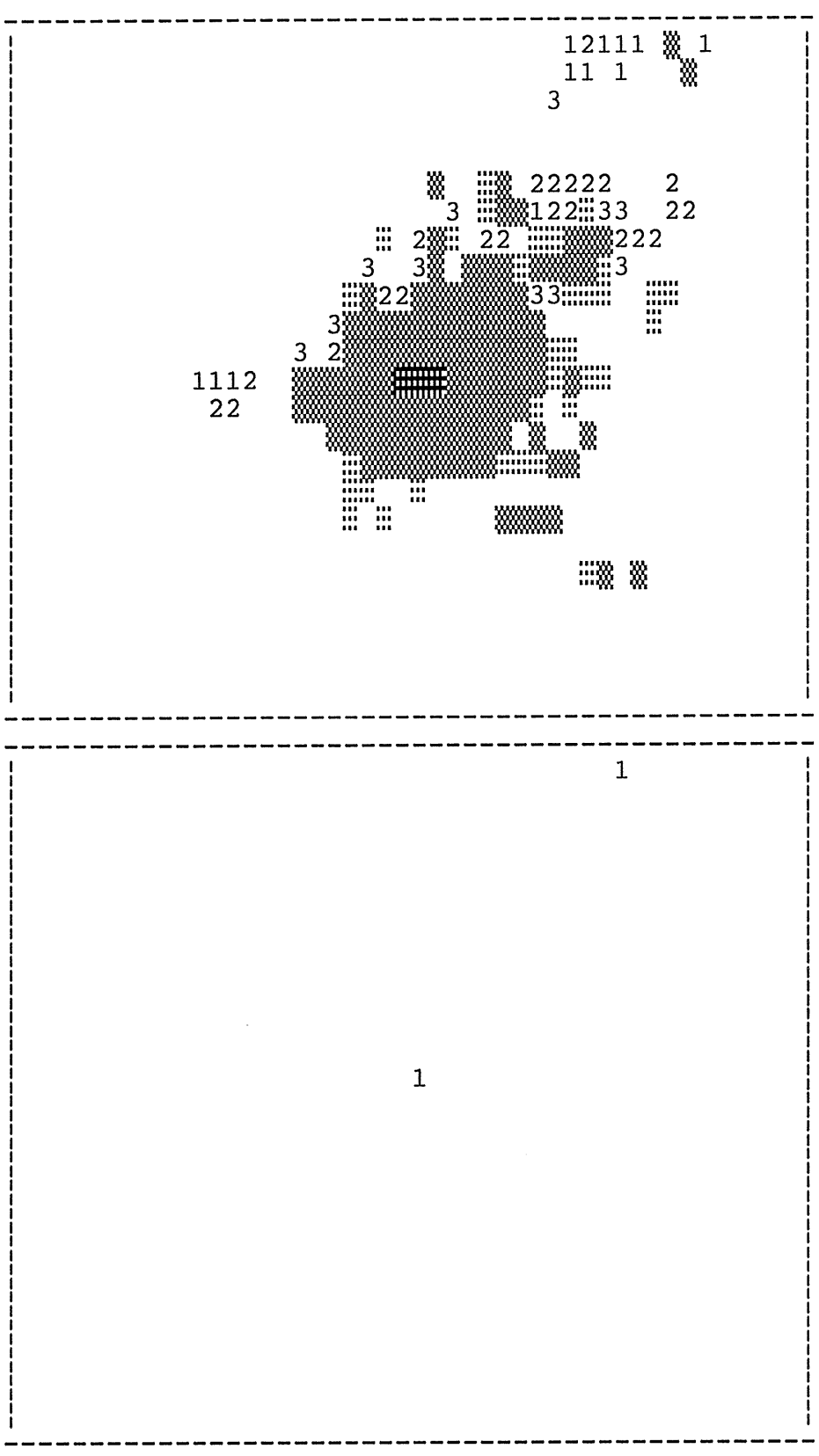
25 aug


111
2
2 11111 1
2 2 111222 11
2 1 2 11 33 33111
2 2 333 3 22
2 1133 3322222 22
23 3 2
2 1 322
1 3 2 32
11 333 2 2
33 3
2 3 3 233
22 2
2 2
3
1
1 1
1
1
1
1

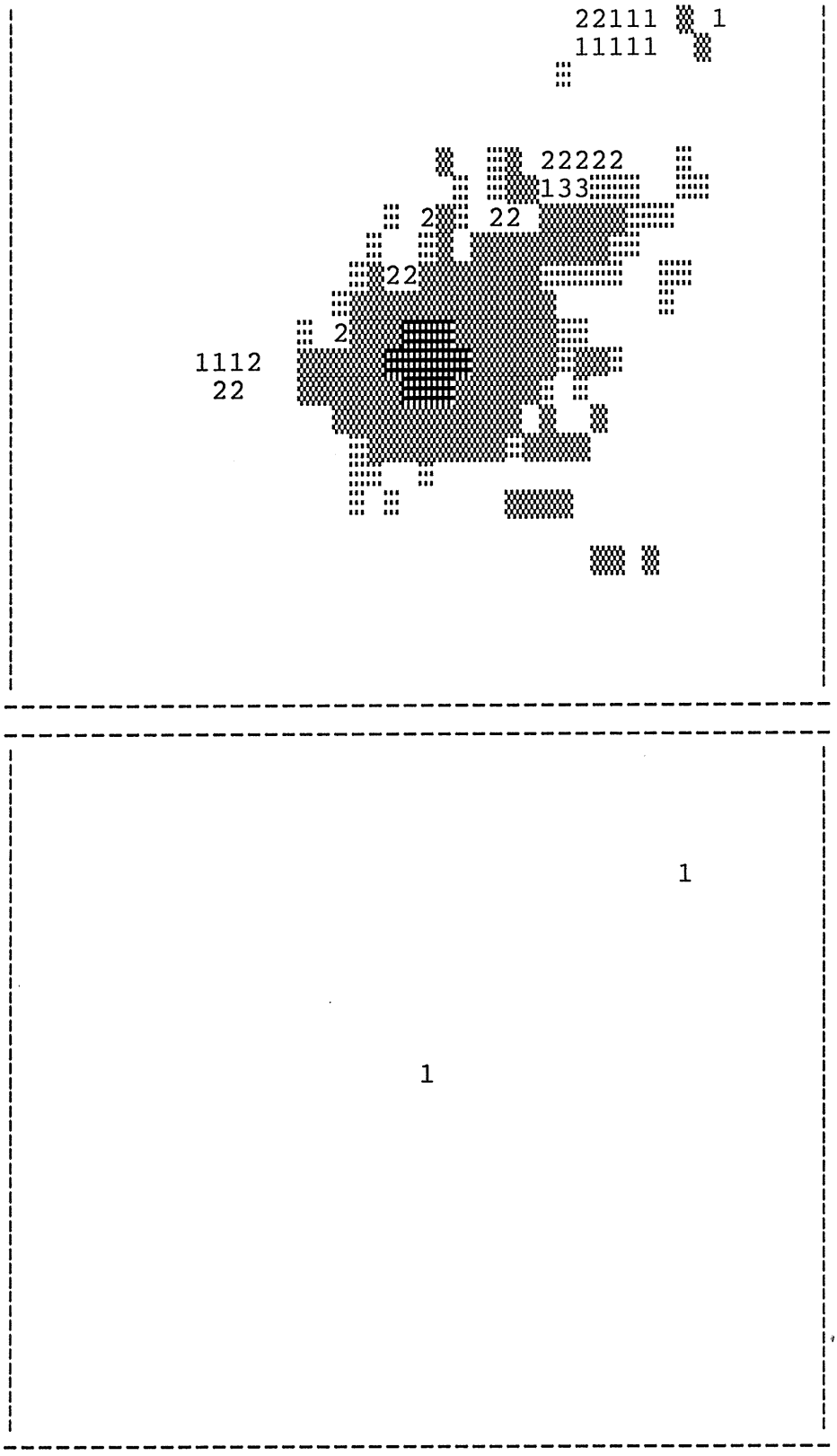

1 sept

8 sept

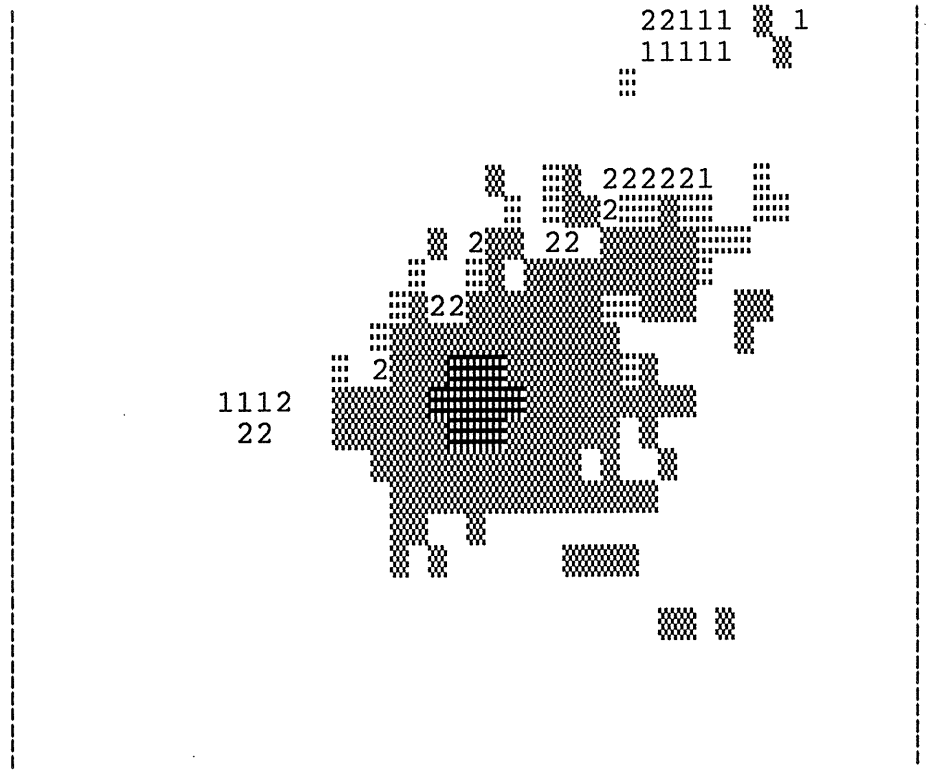

1111 1
11 1
2
3 11111 2
3 3 122322 22
3 2 3 22 222
3 3 32
3 22 33333 33
3 3
2 1 33
1111 3 3
11 3
3 3
33 3
3 3

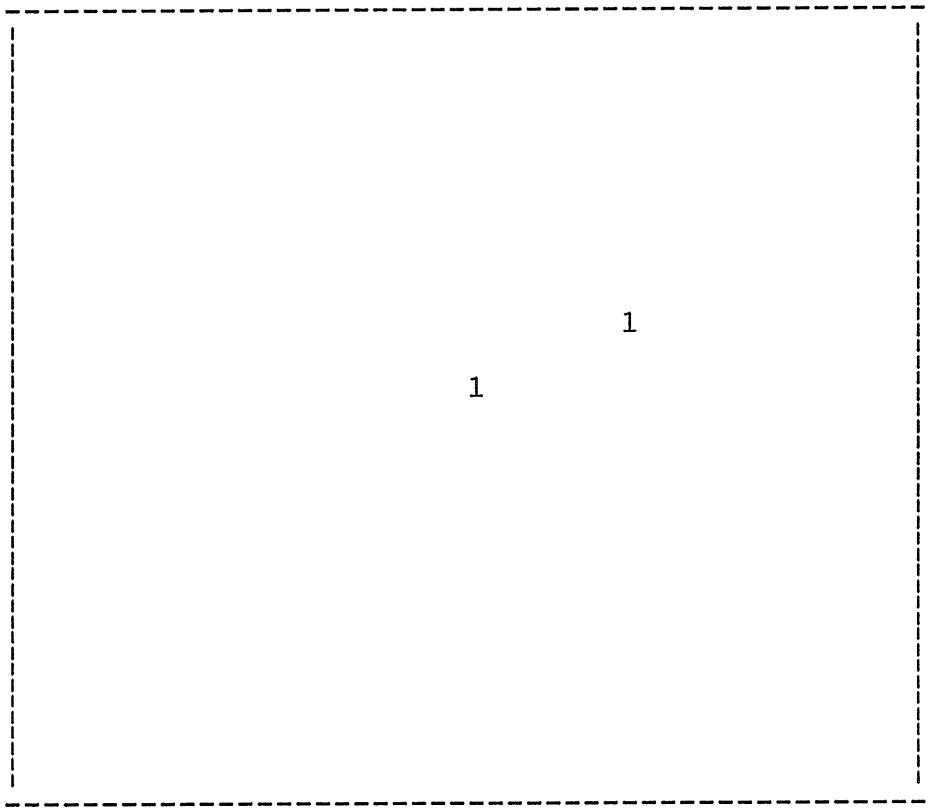

15 sept
12111 1
11 1
3
22222 2
3 122 33 22
2 22 222
3 3 3
22 33
3
3 2
1112
22
1
1

22 sept

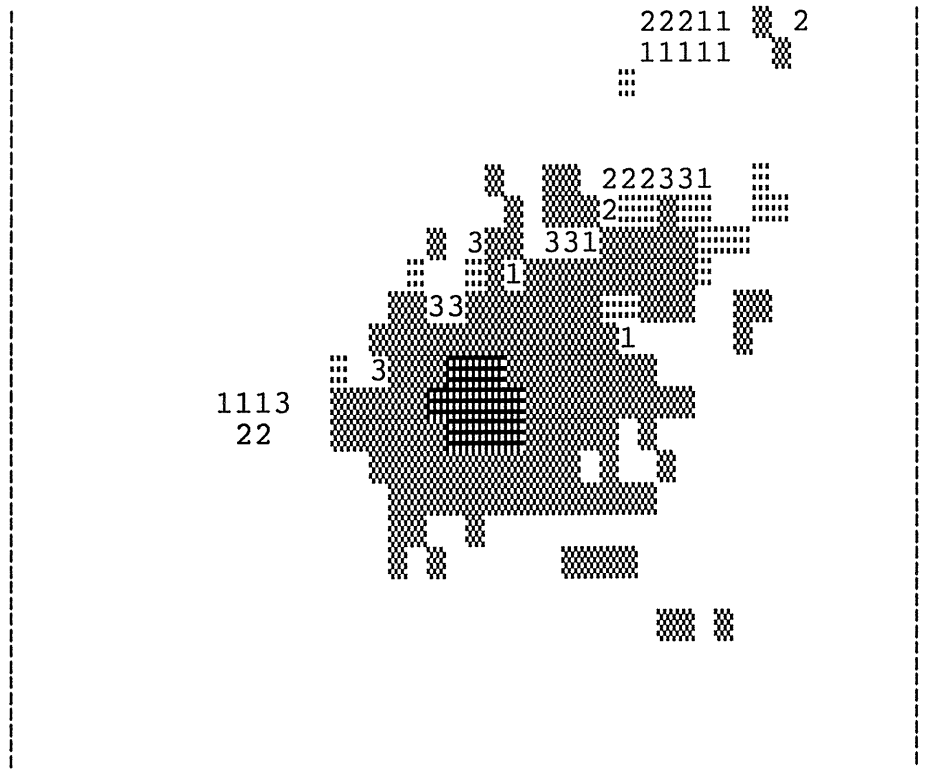


29 sept

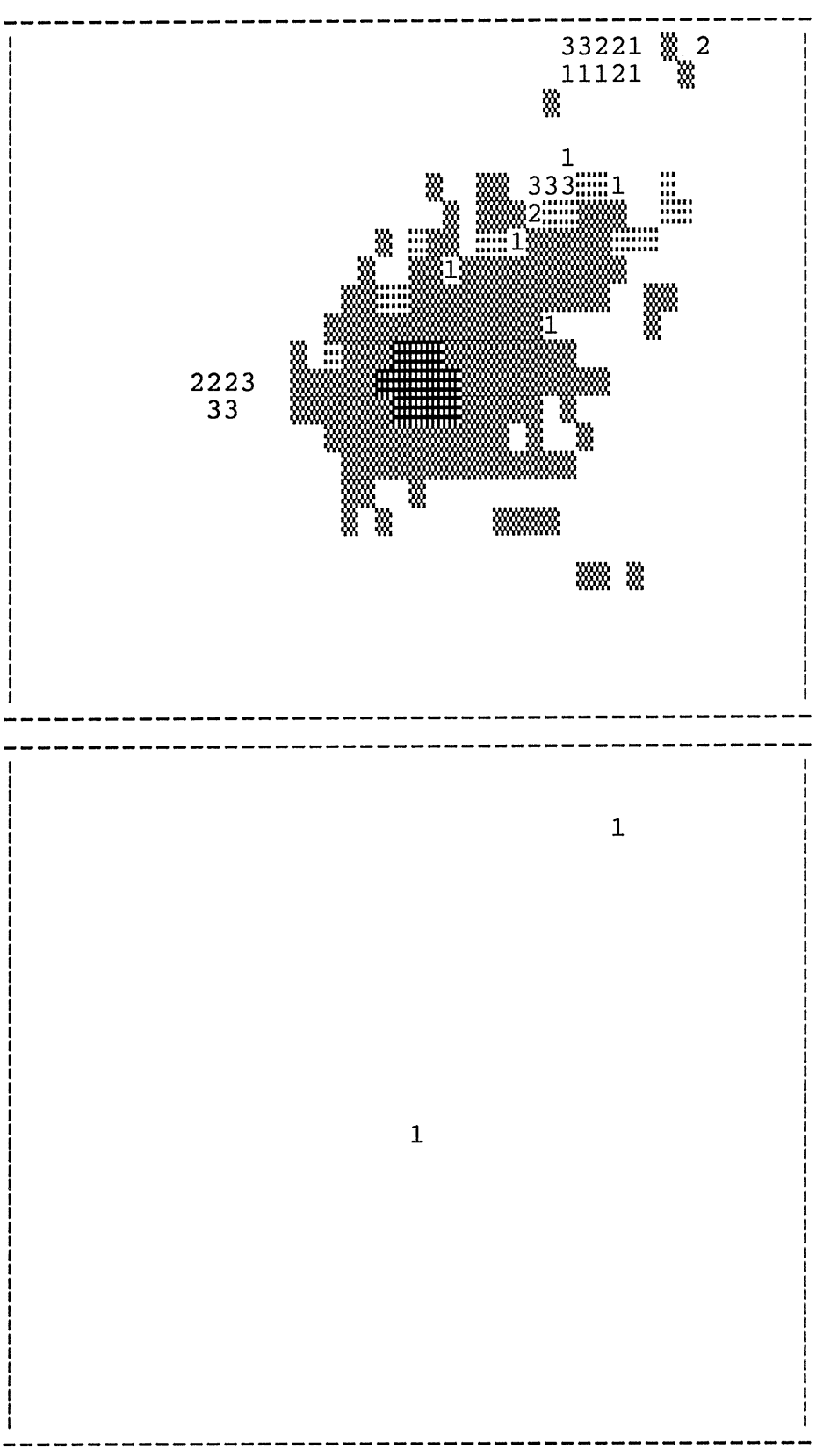




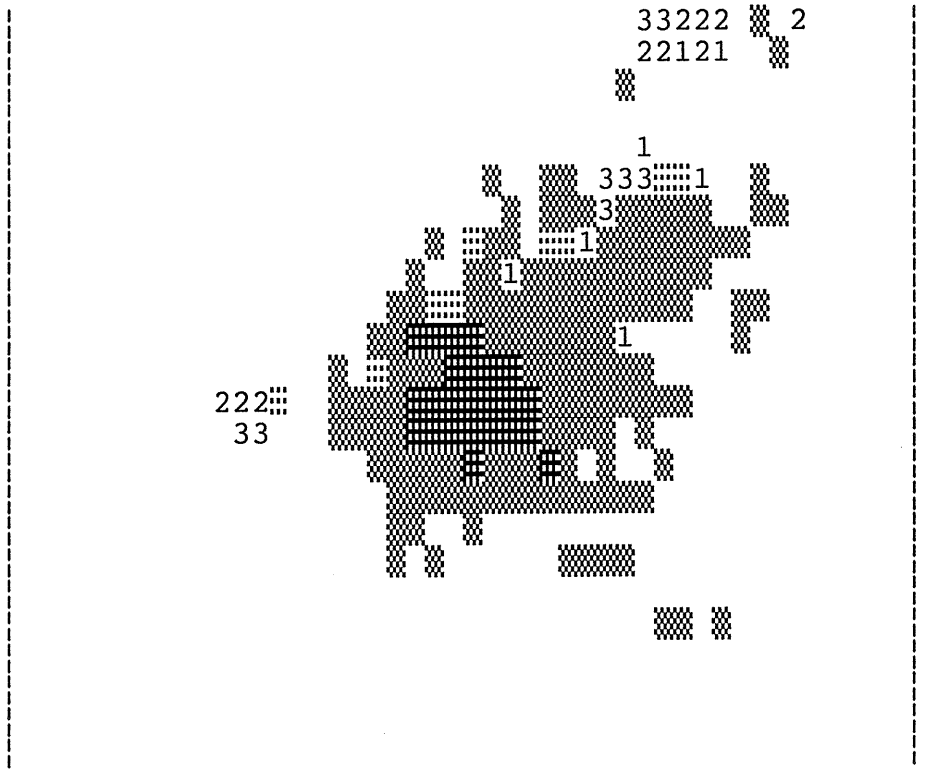
6 okt



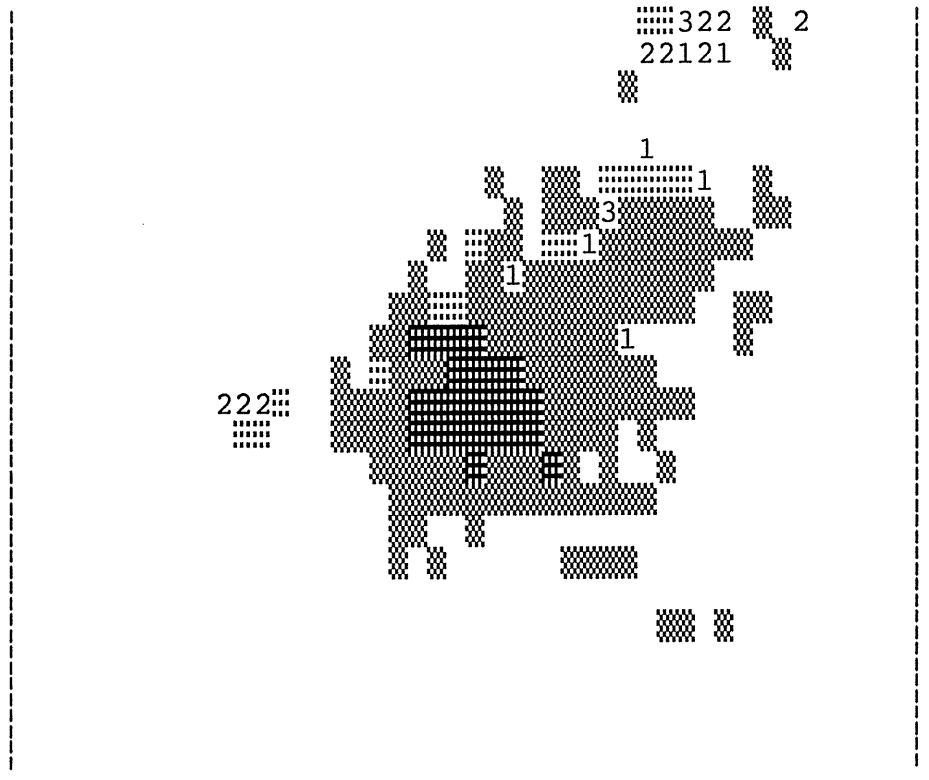
1


13 okt
33221 2
11121
1
333 1
2
1
1
1
2223
33
1
1

20 okt

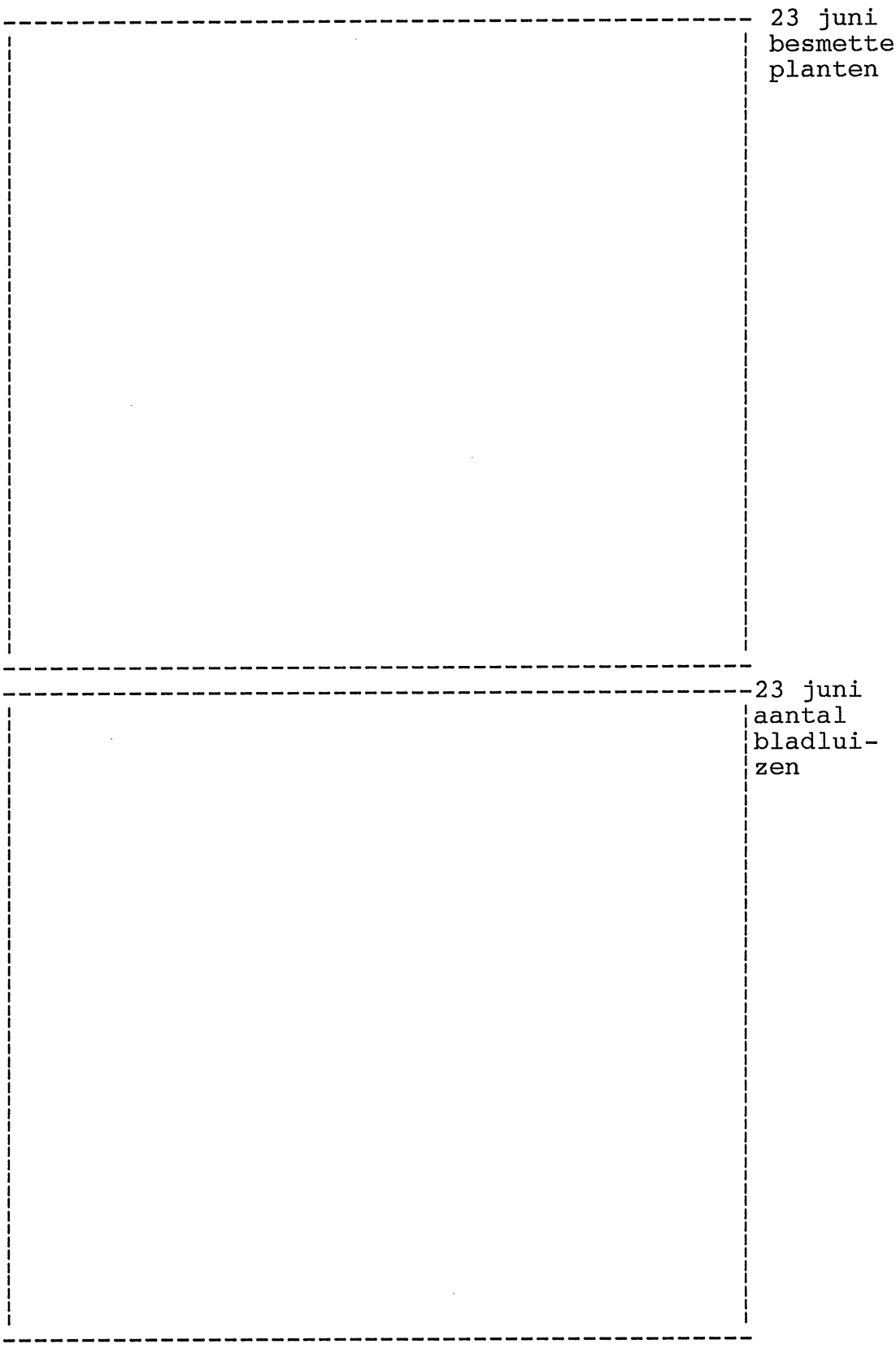


27 okt

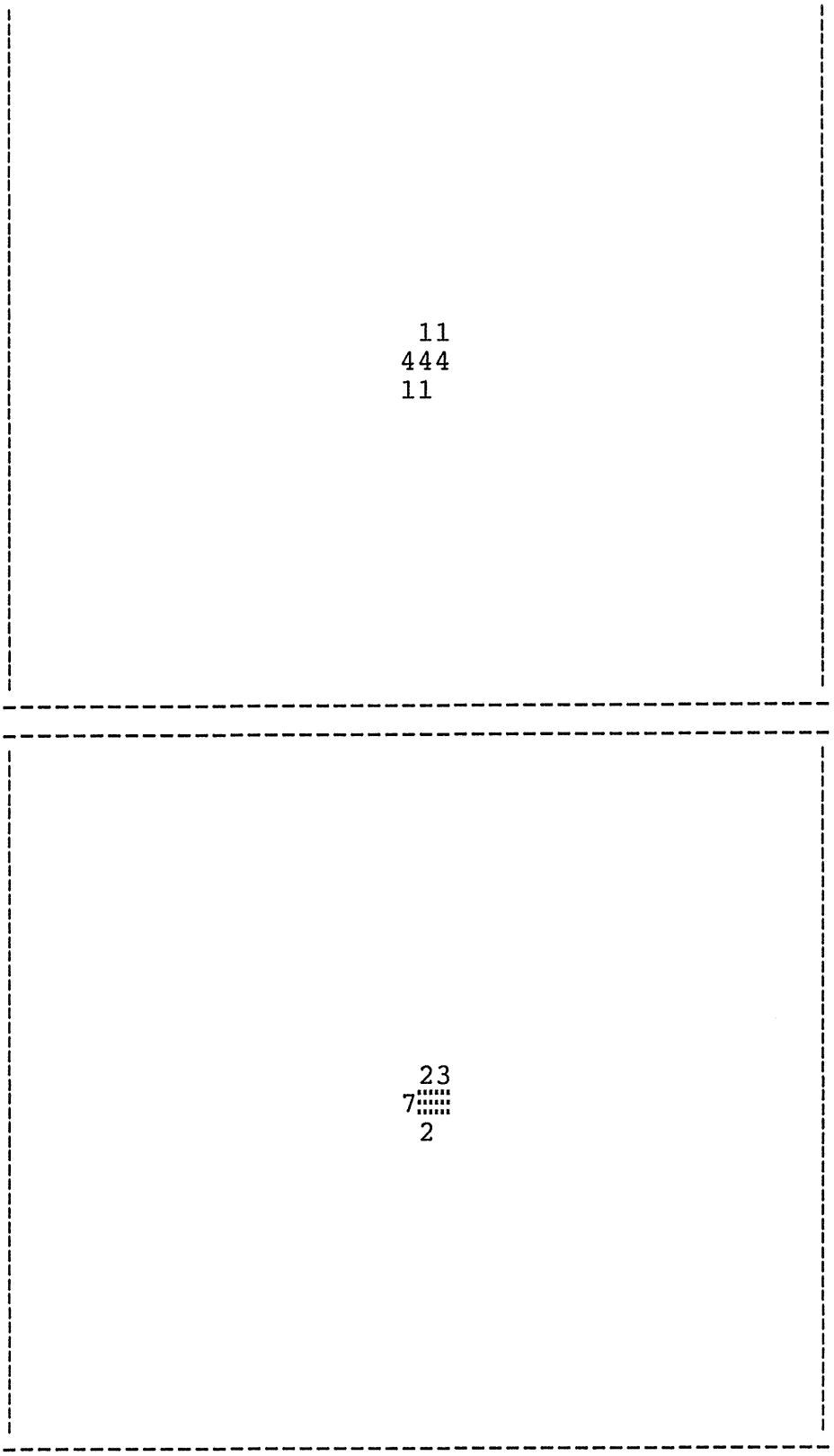


Bijlage B

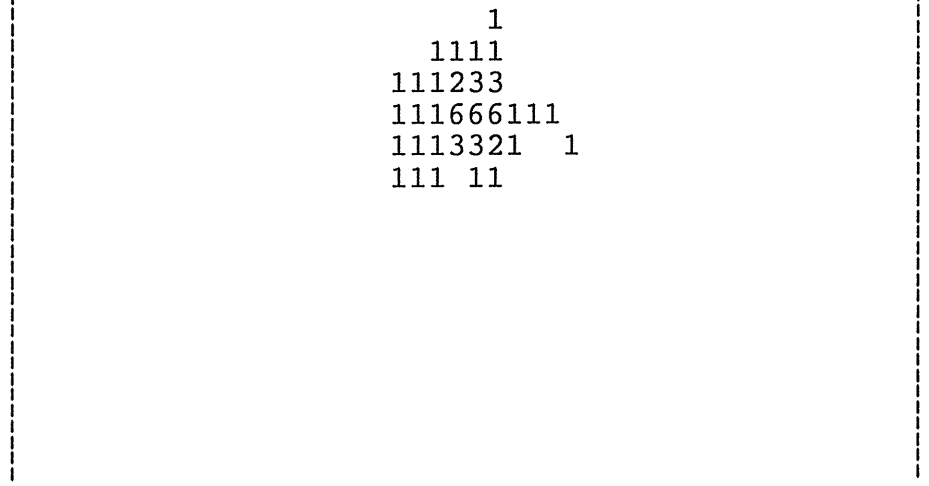
Een simulatie met laatgezaaide bieten (20 mei 1986) met de luizentellingen uit file luizen5.dat.

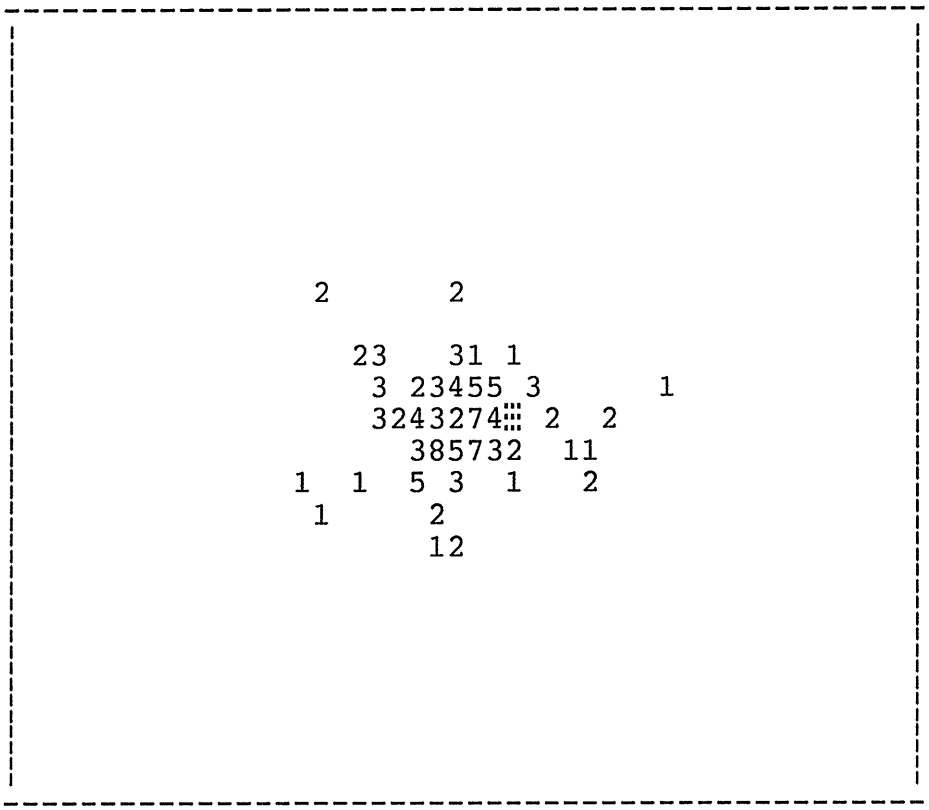


30 juni

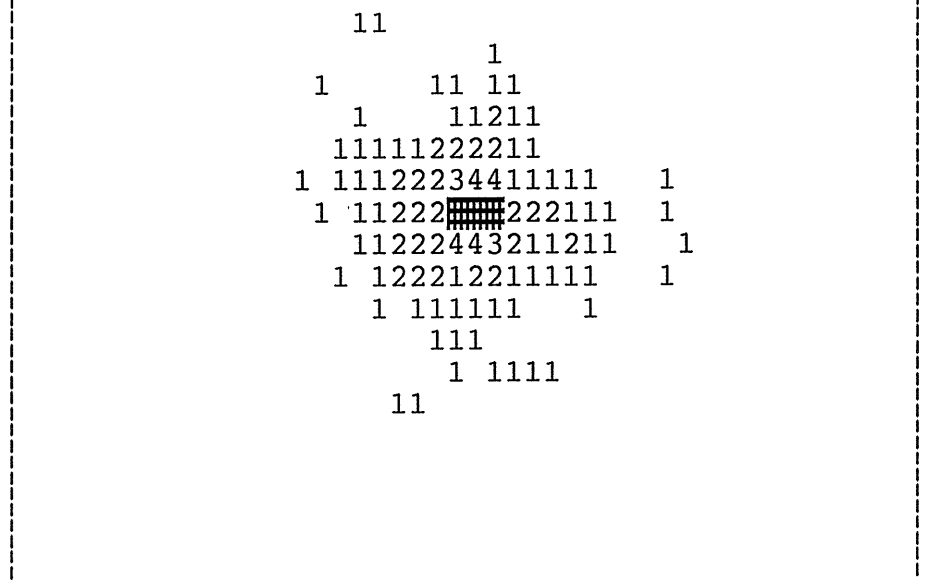


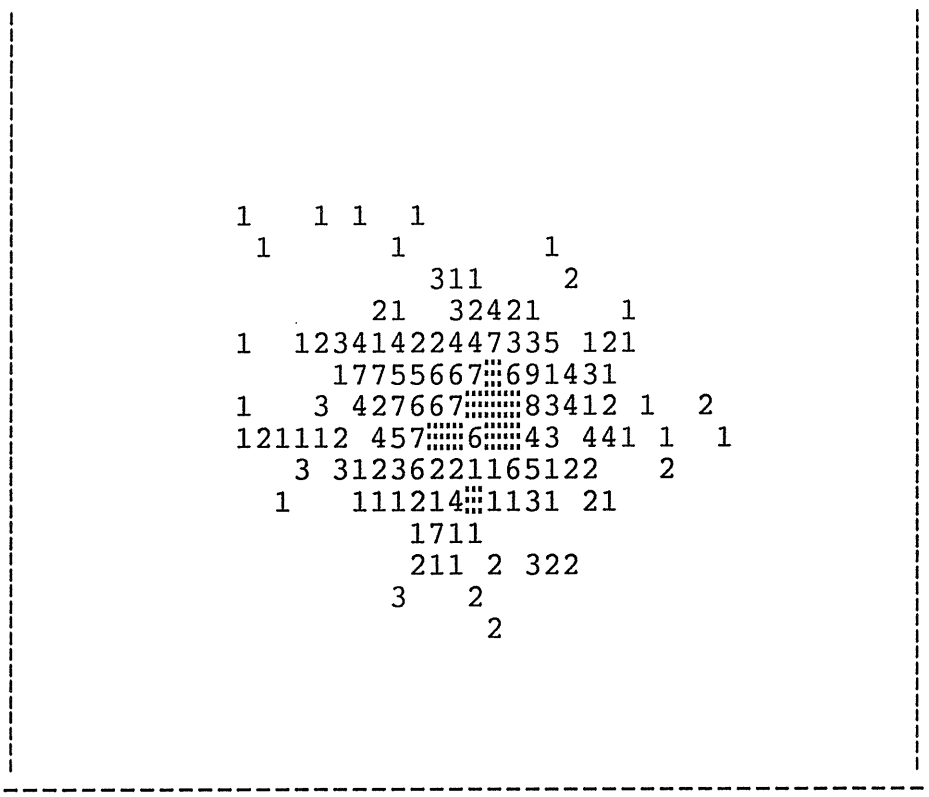
7 juli



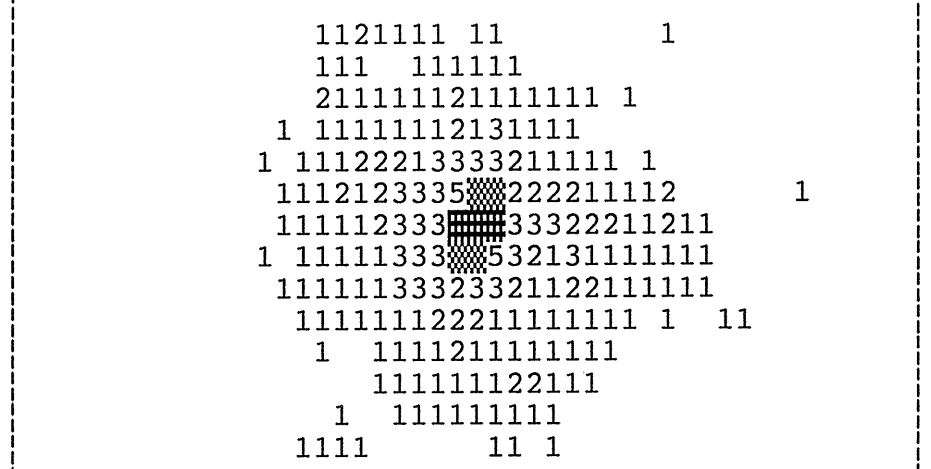


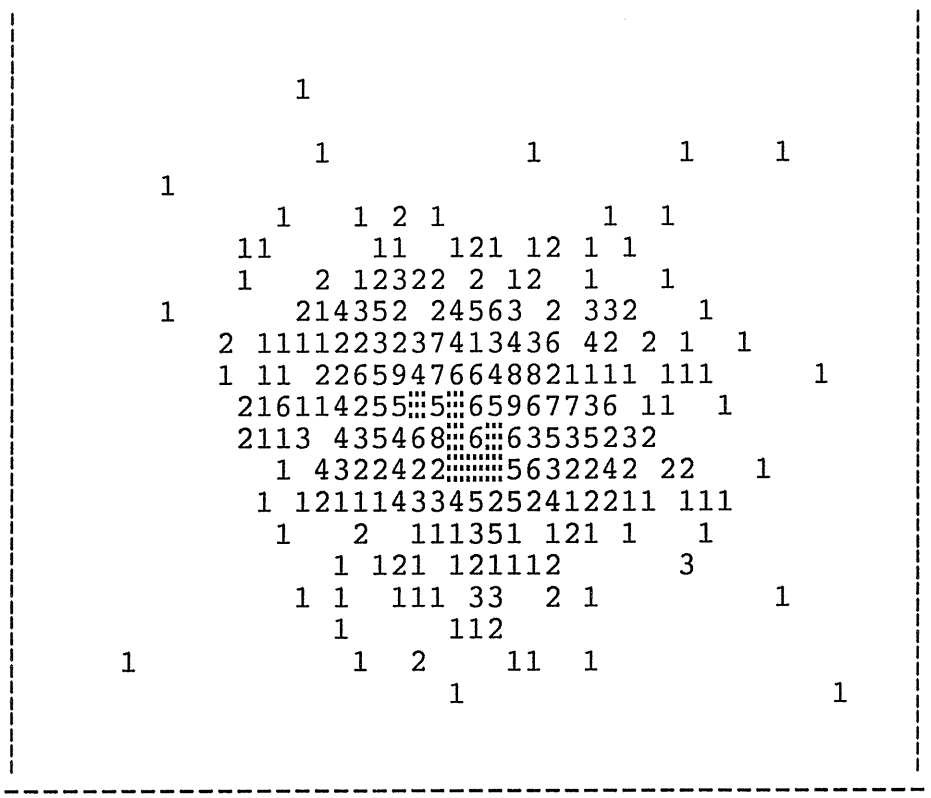
14 juli



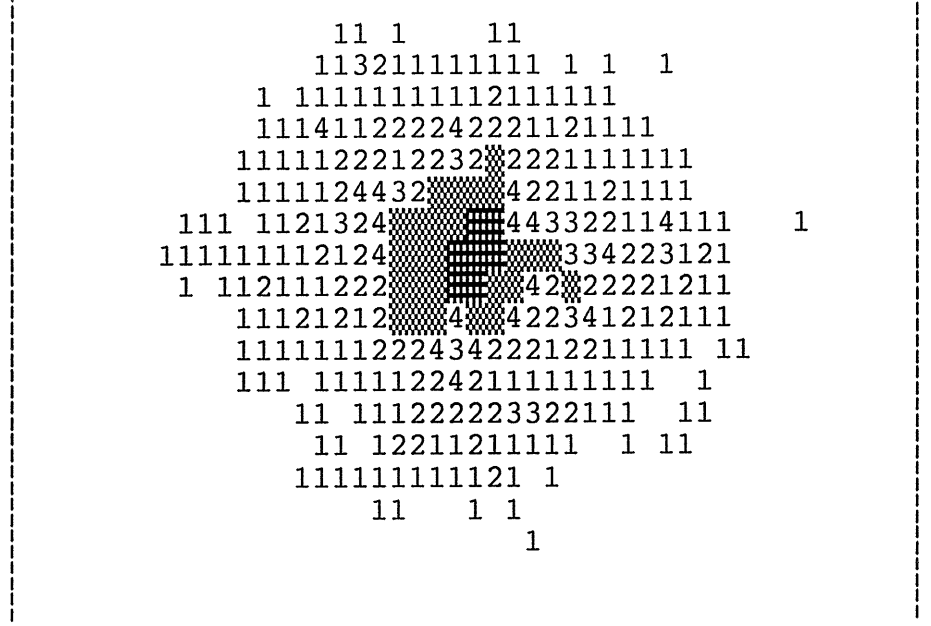


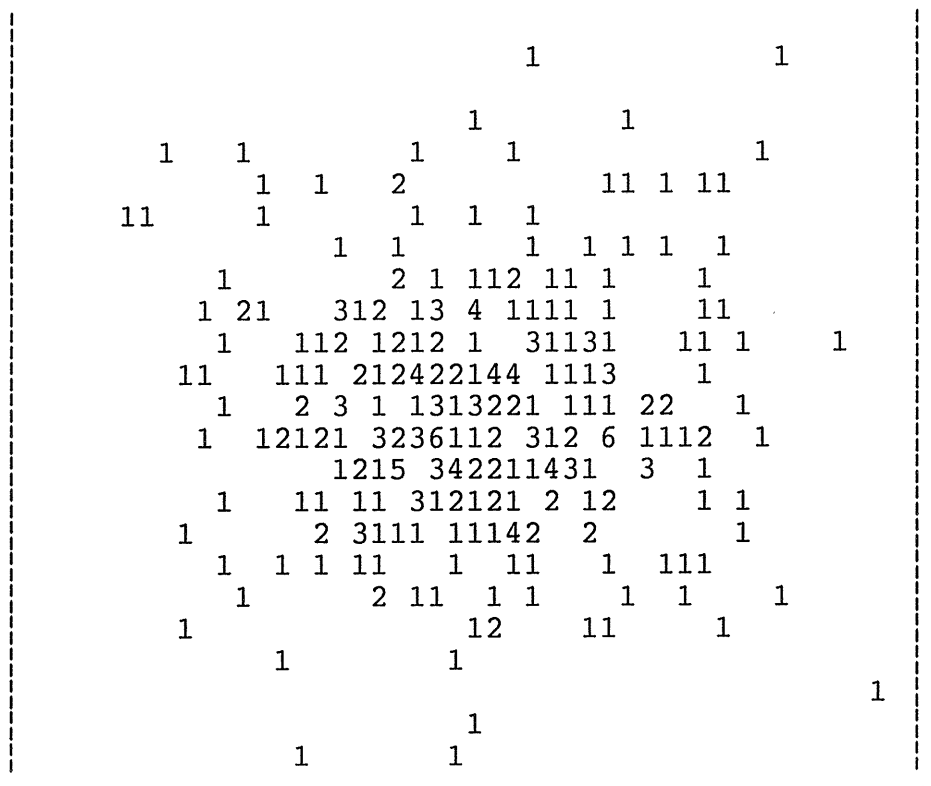
21 juli



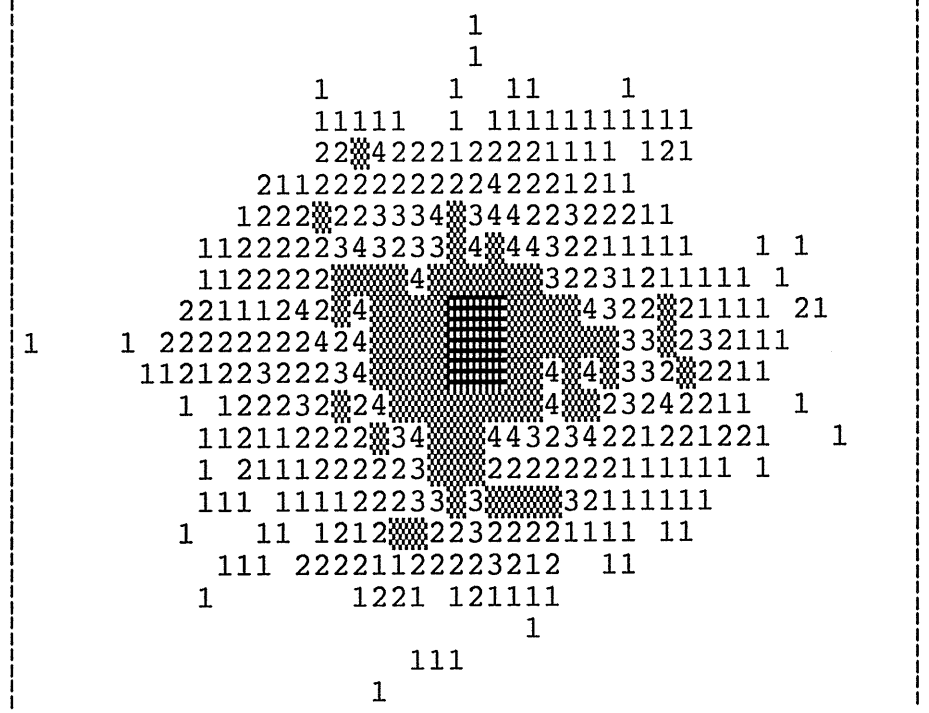


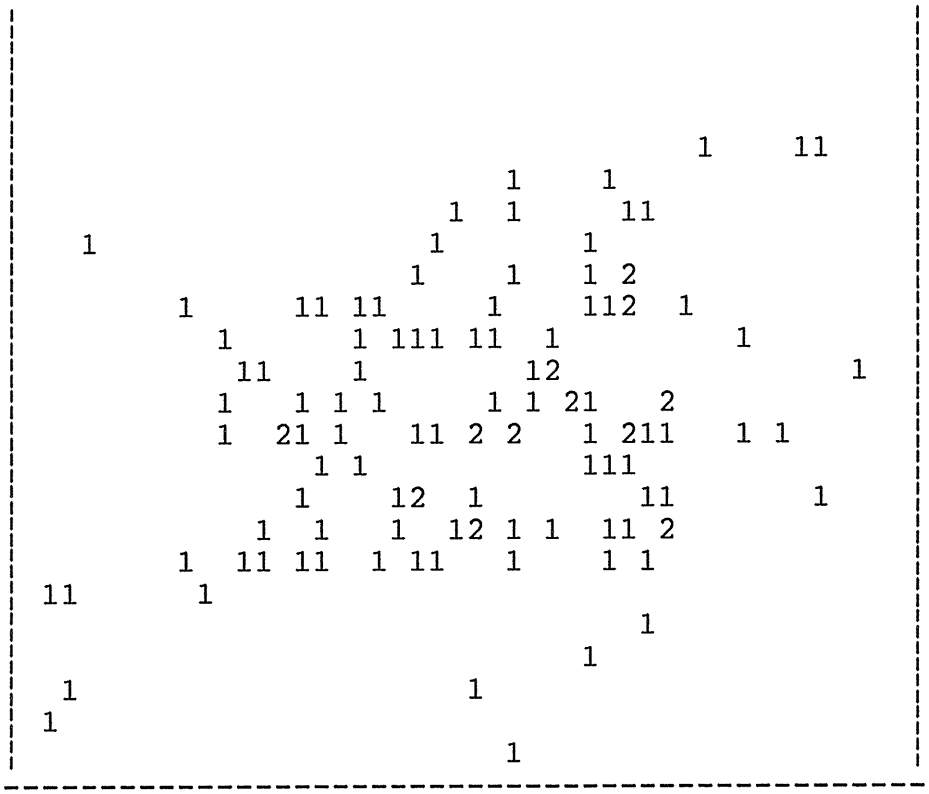
28 juli



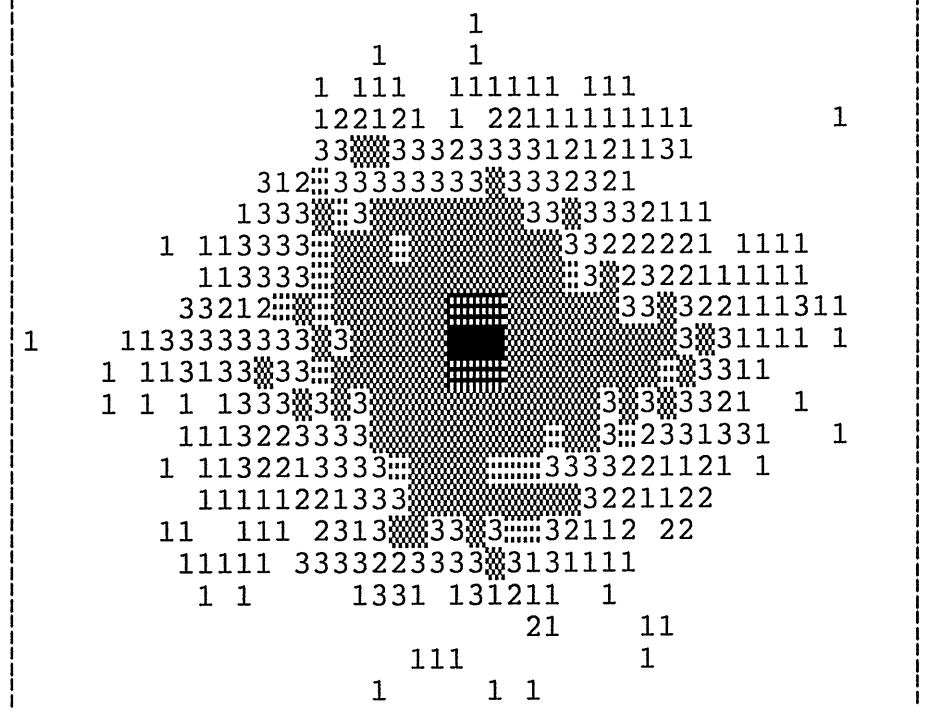


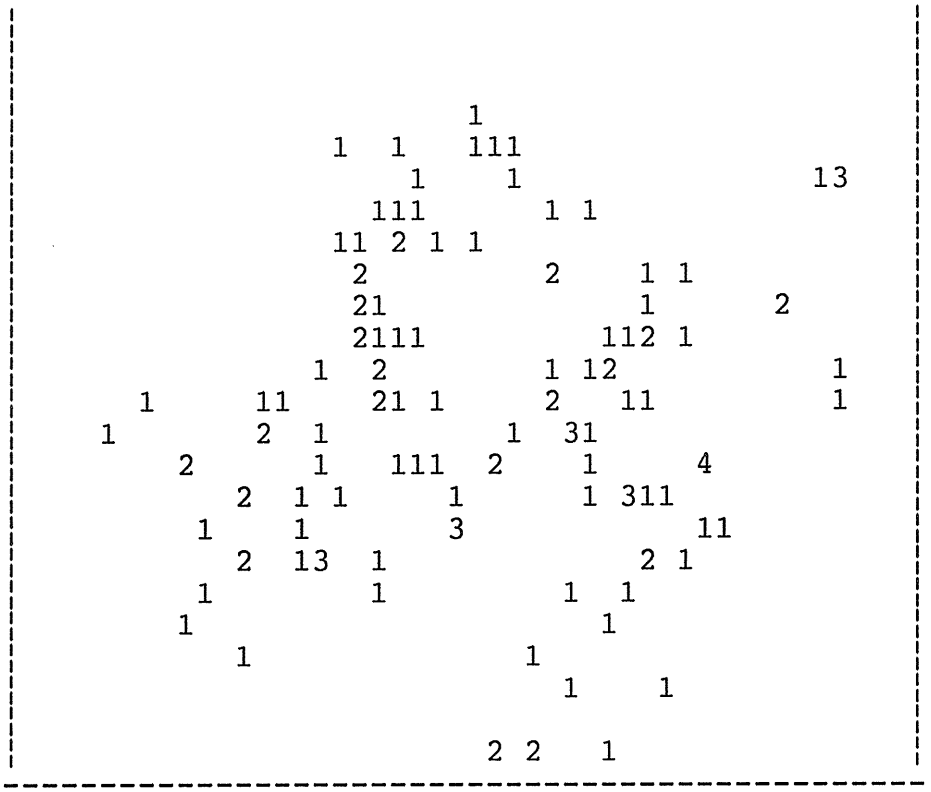
4 aug



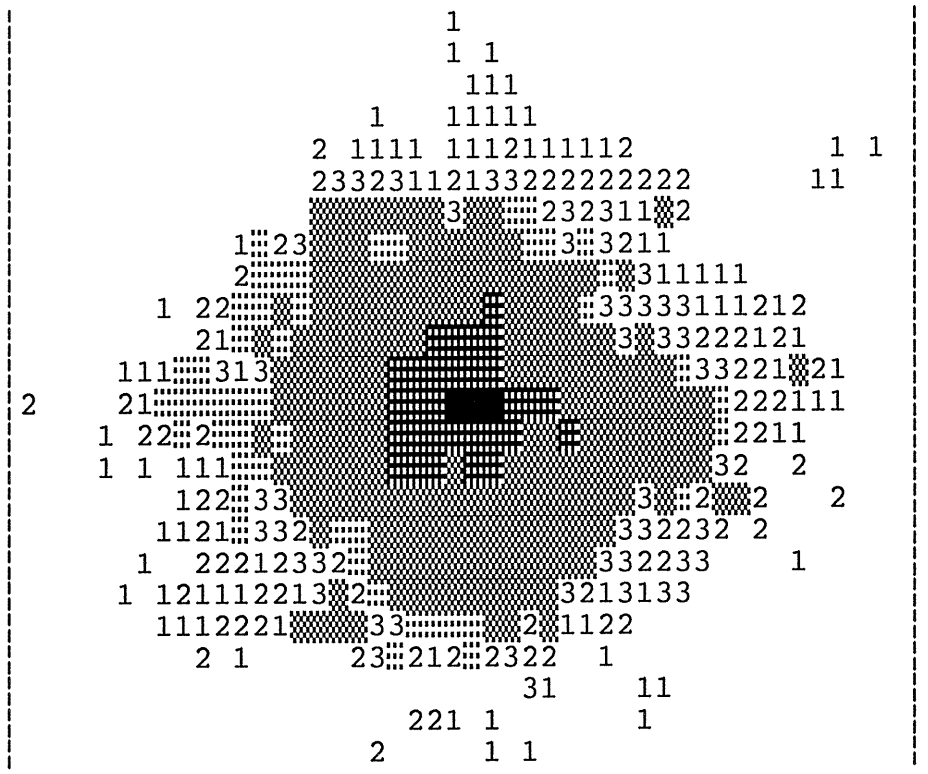


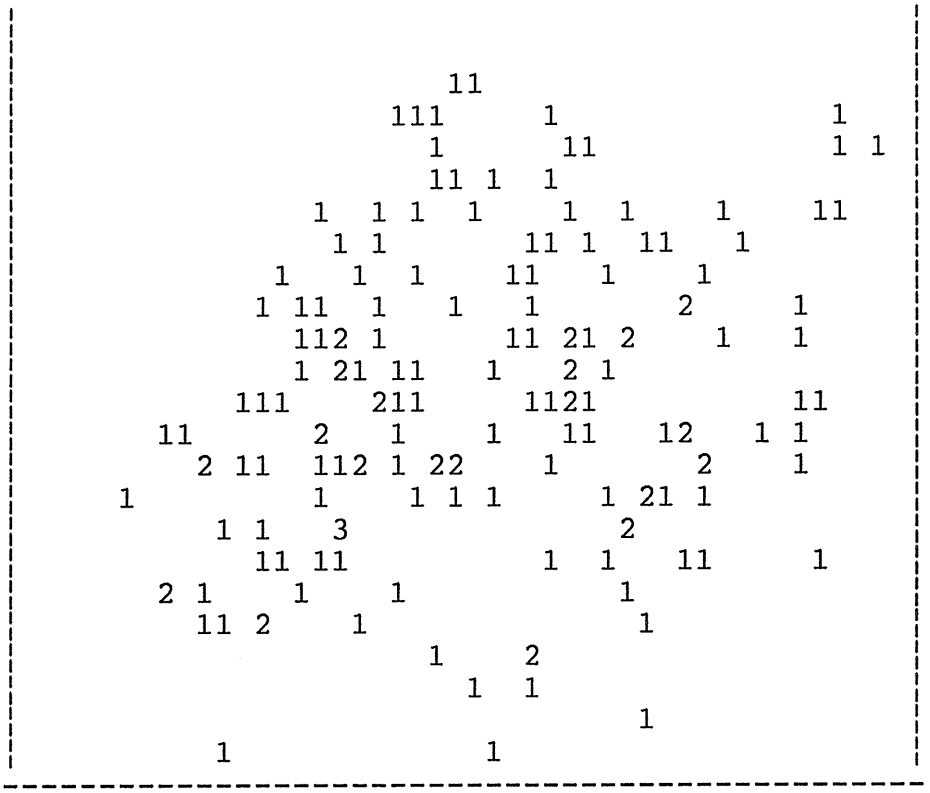
11 aug



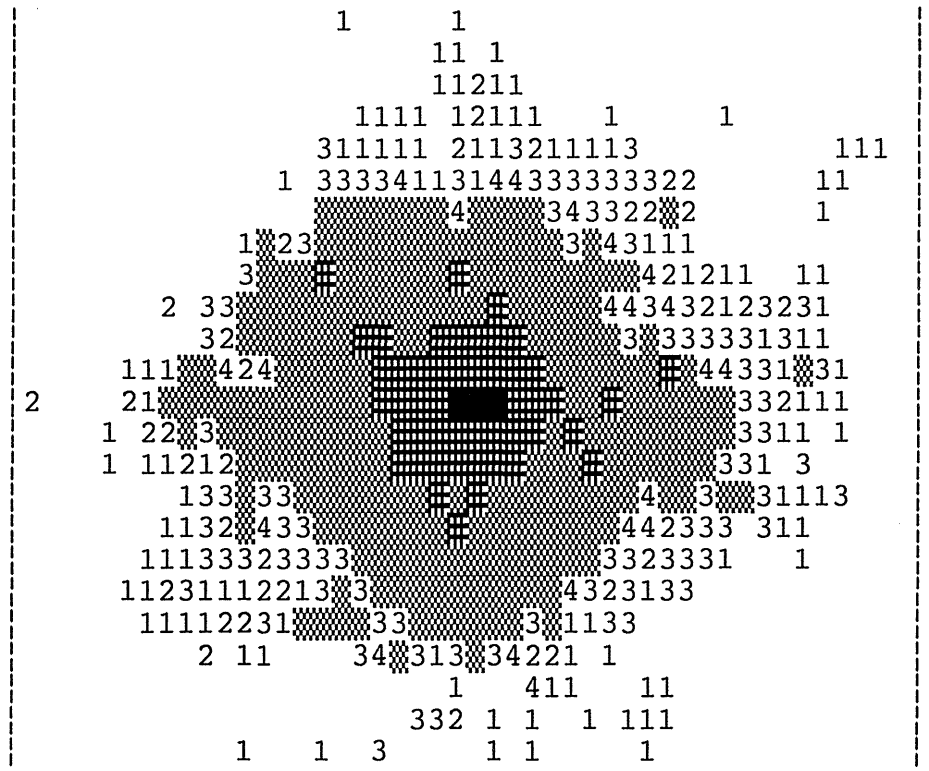


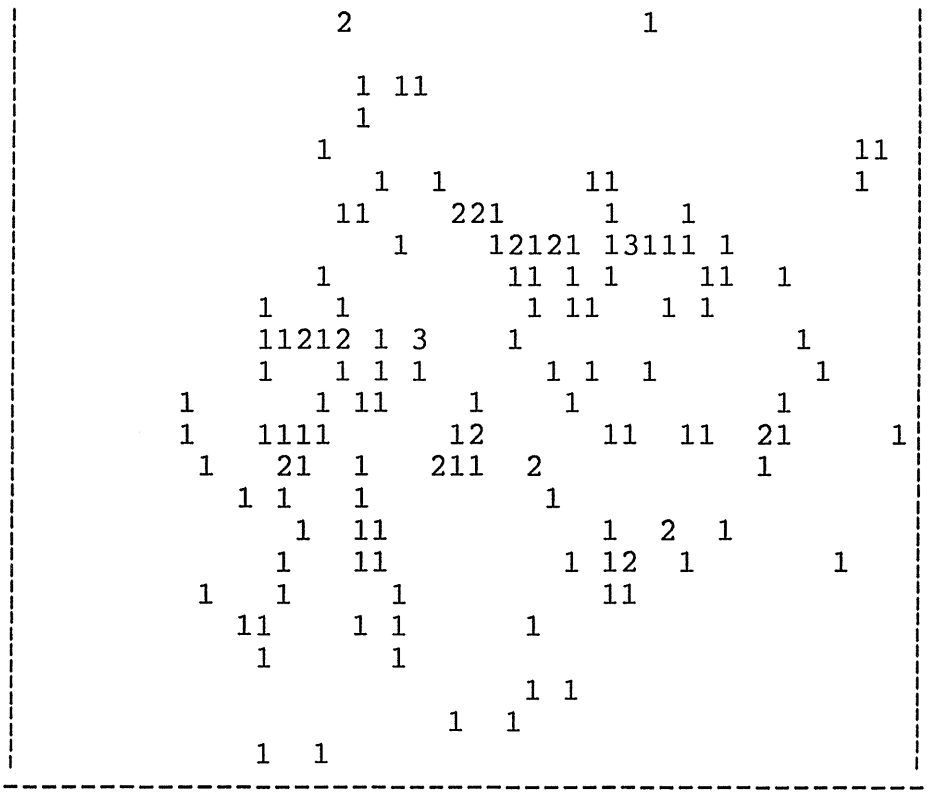
18 aug



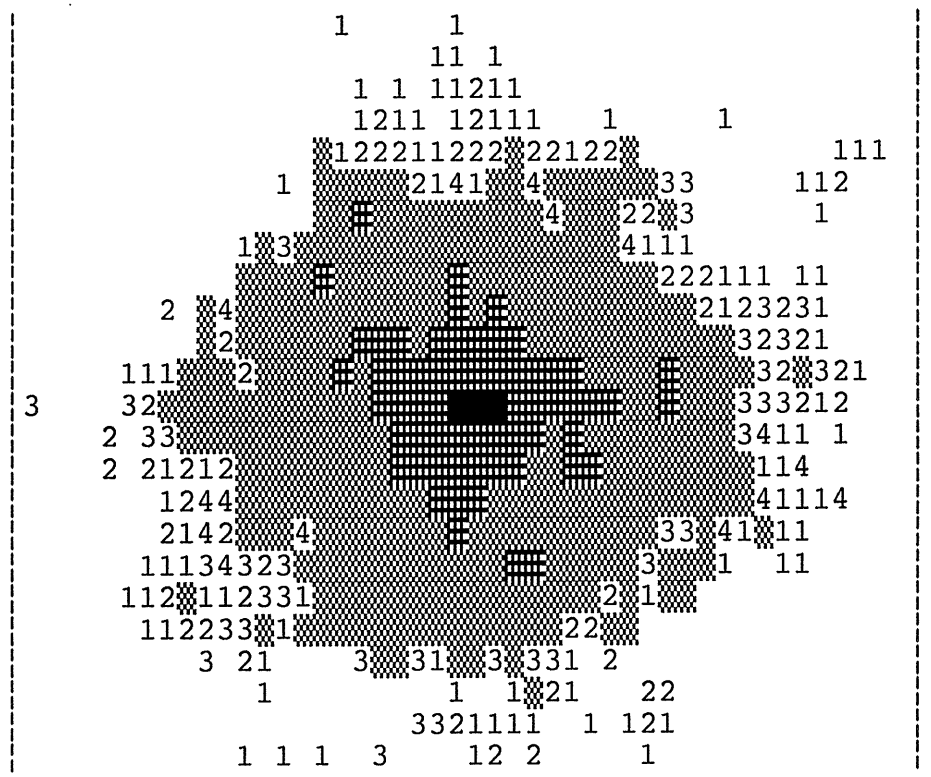


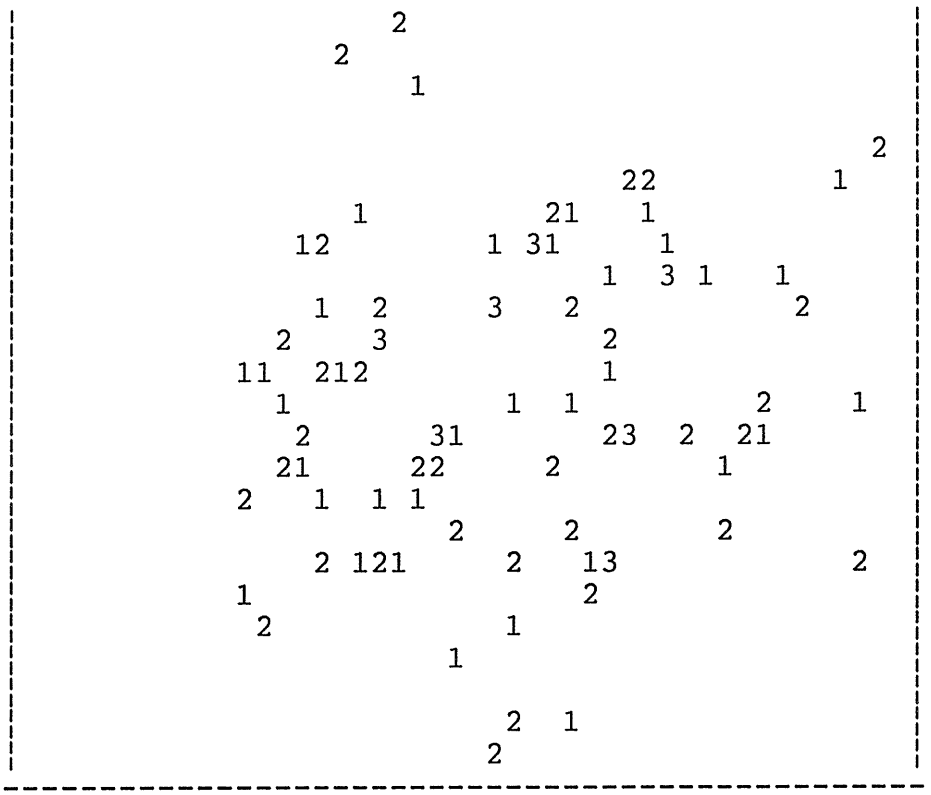
25 aug



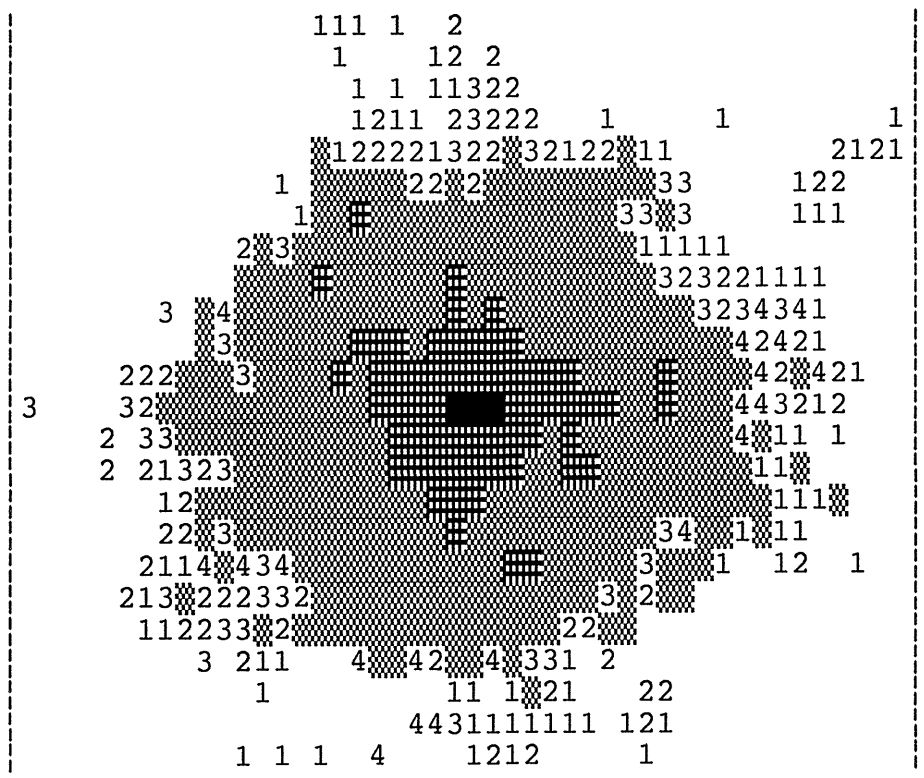


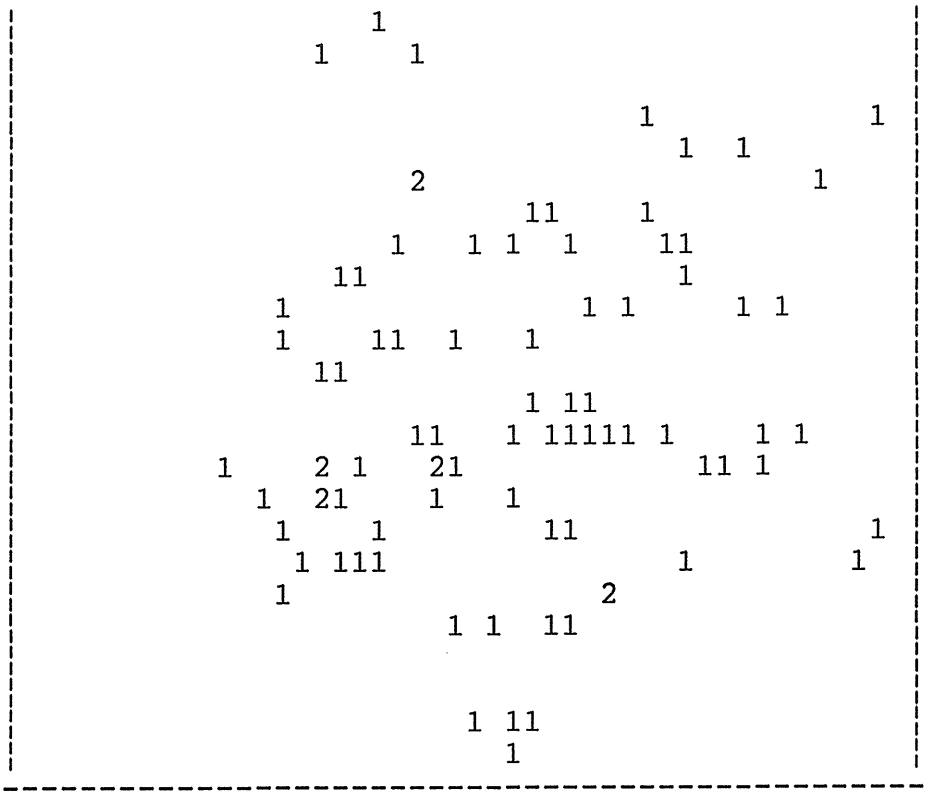
1 sept



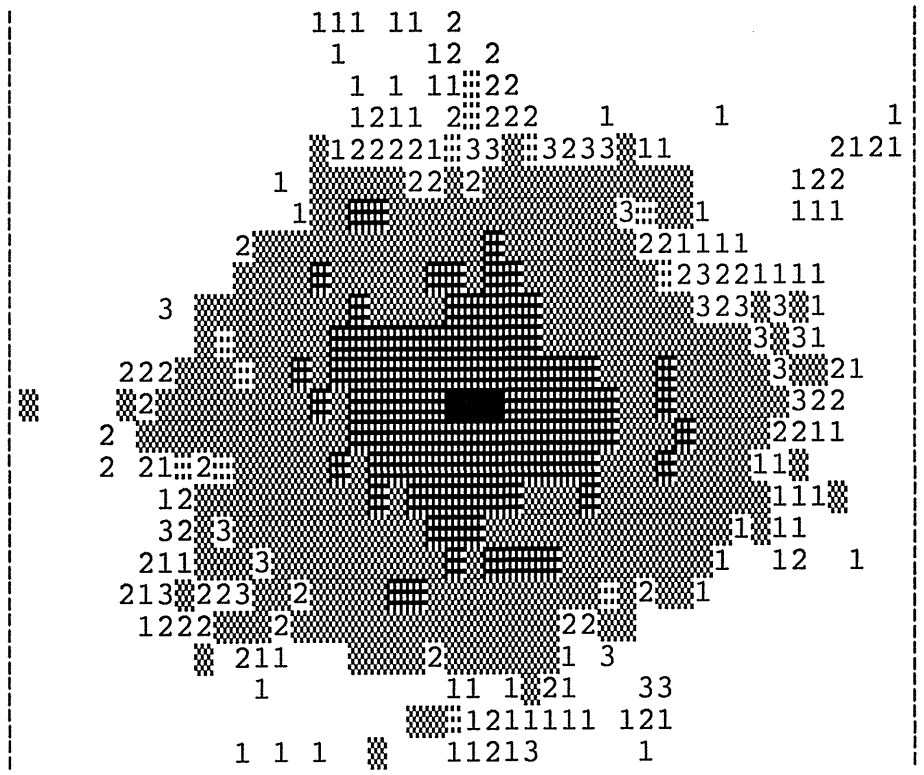


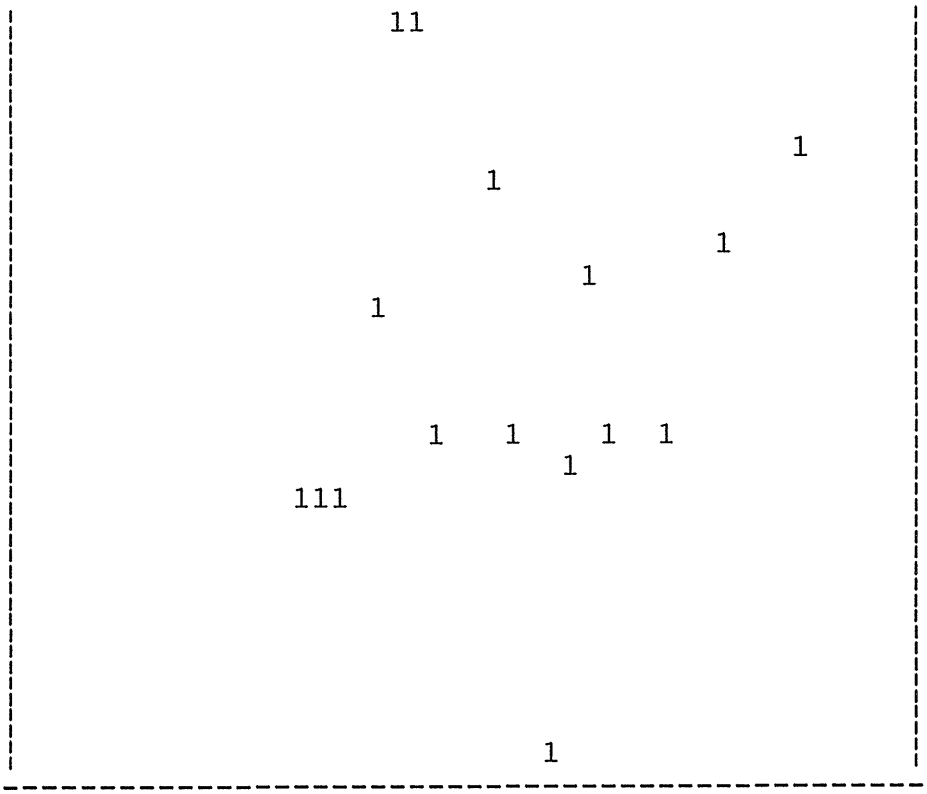
8 sept



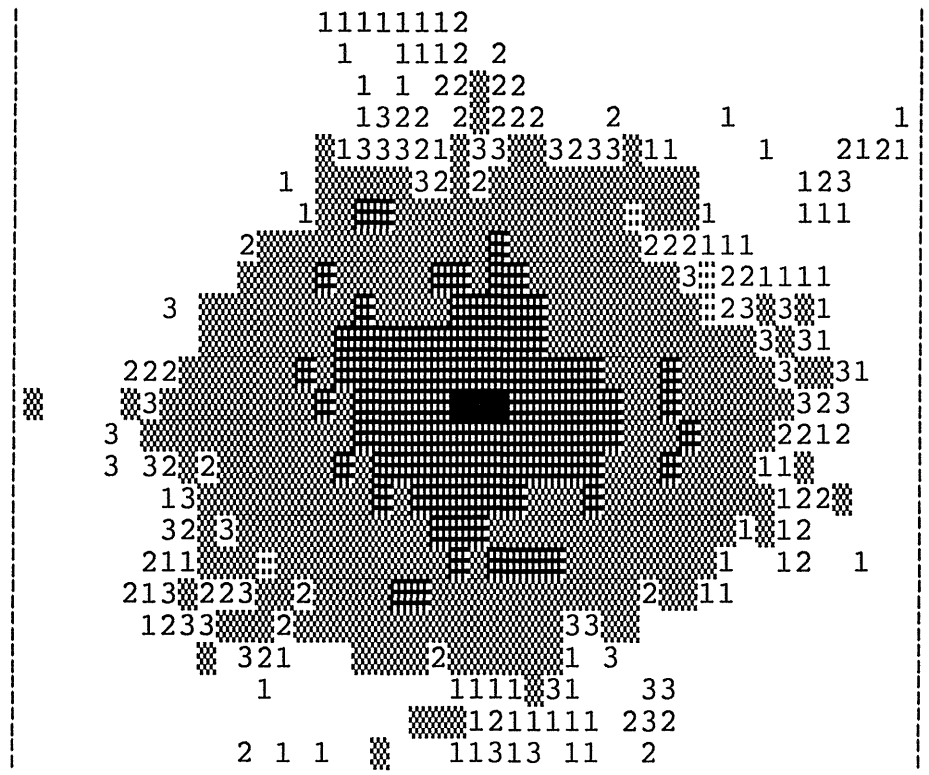


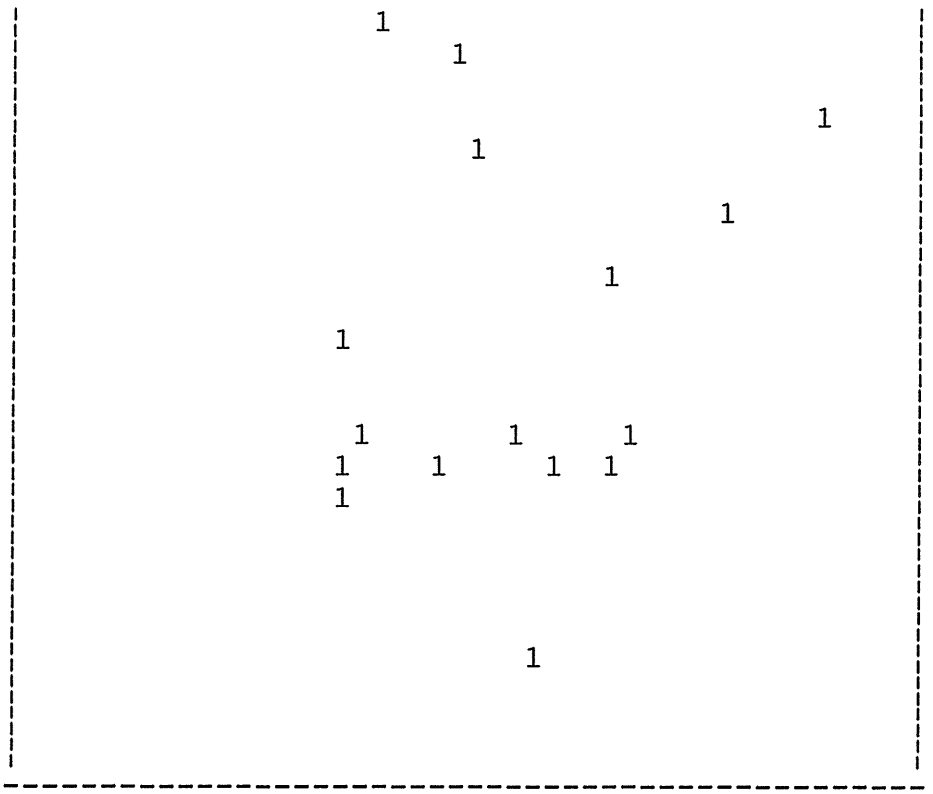
15 sept



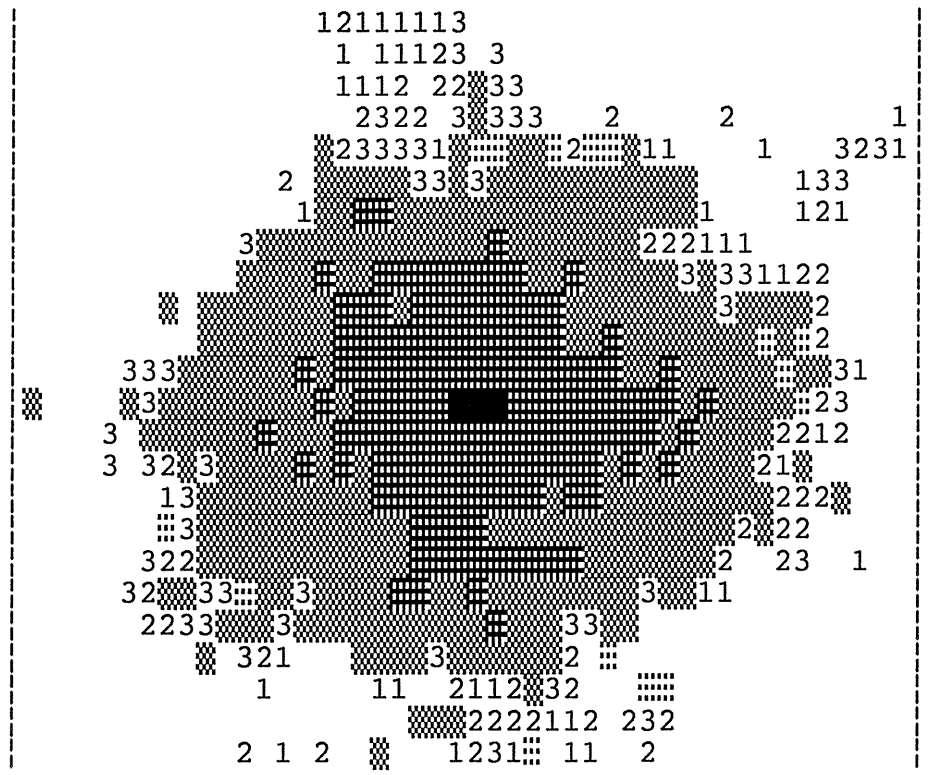


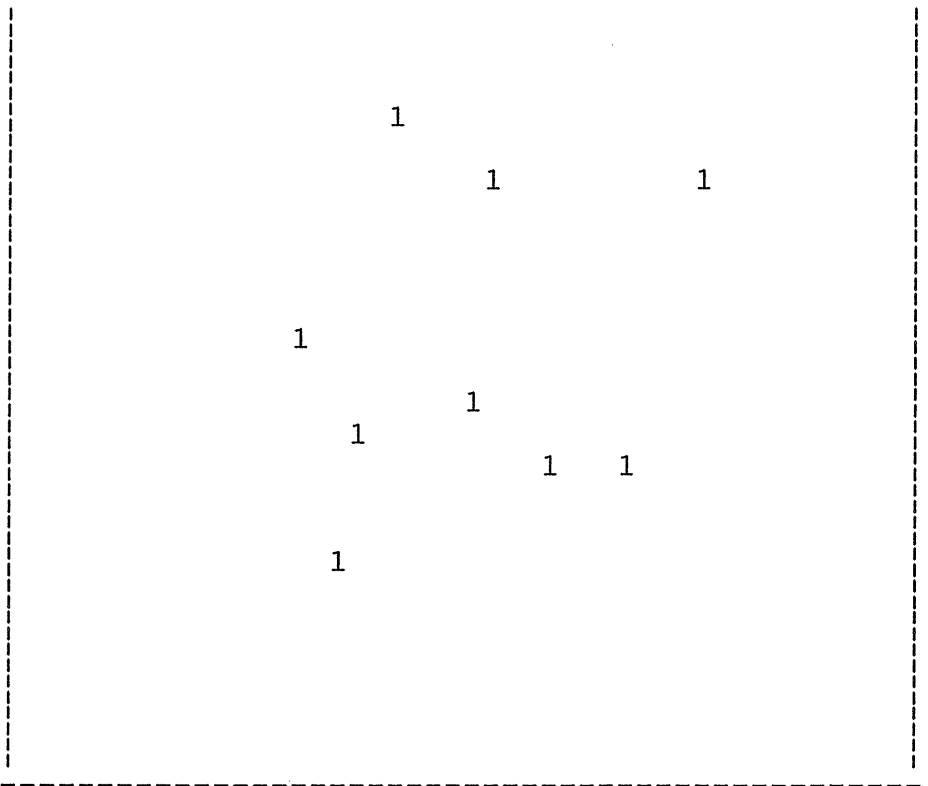
22 sept



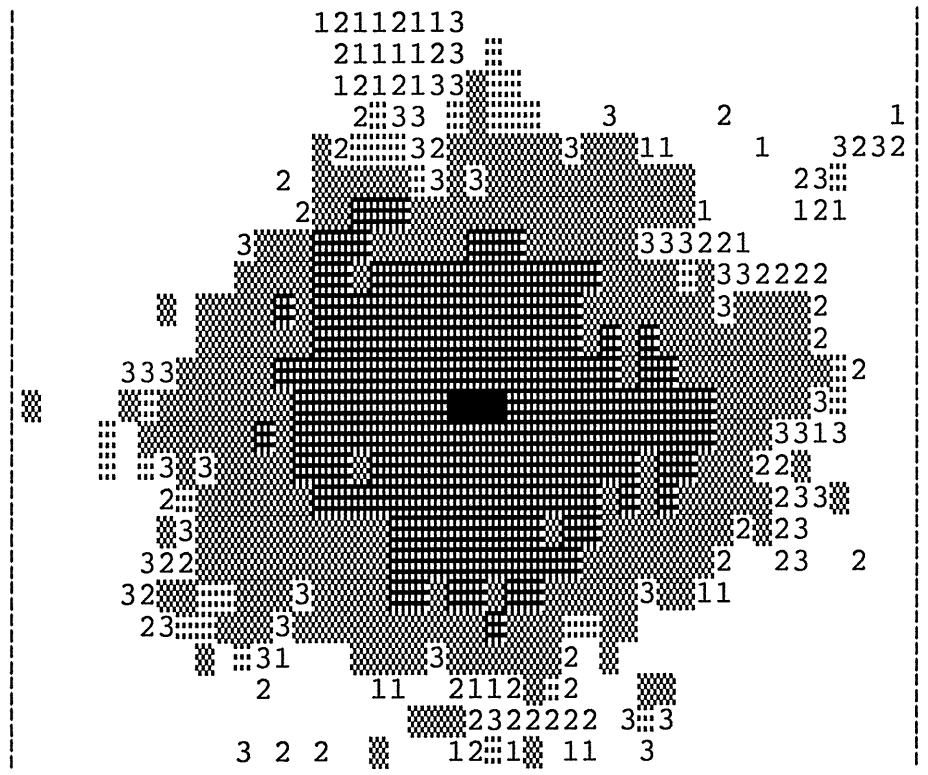


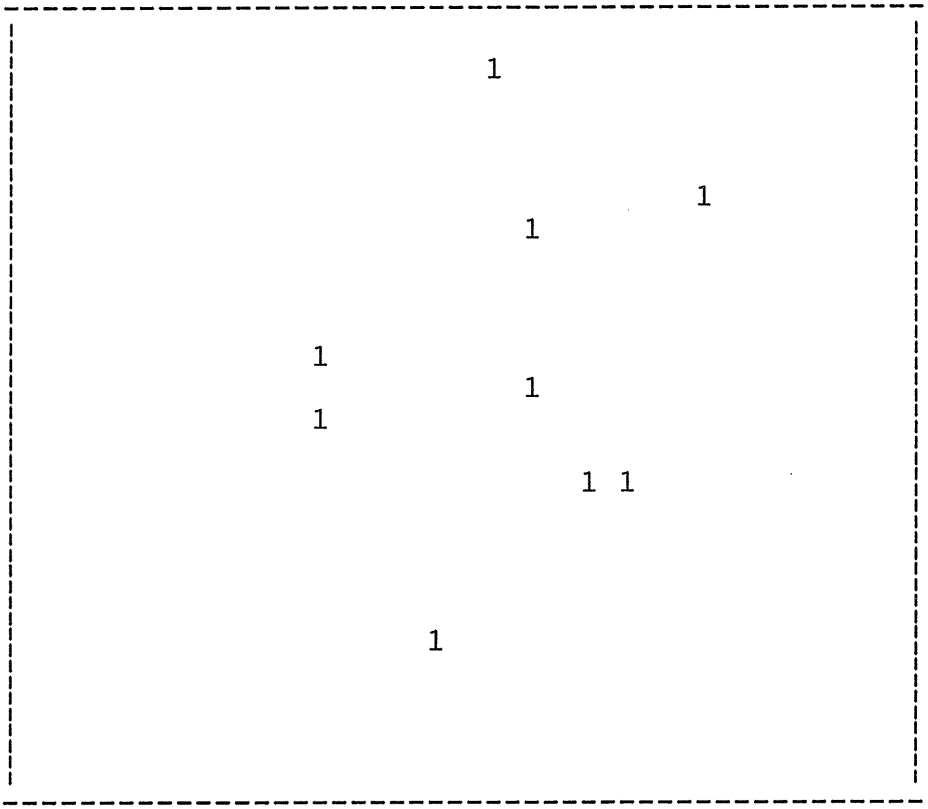
29 sept



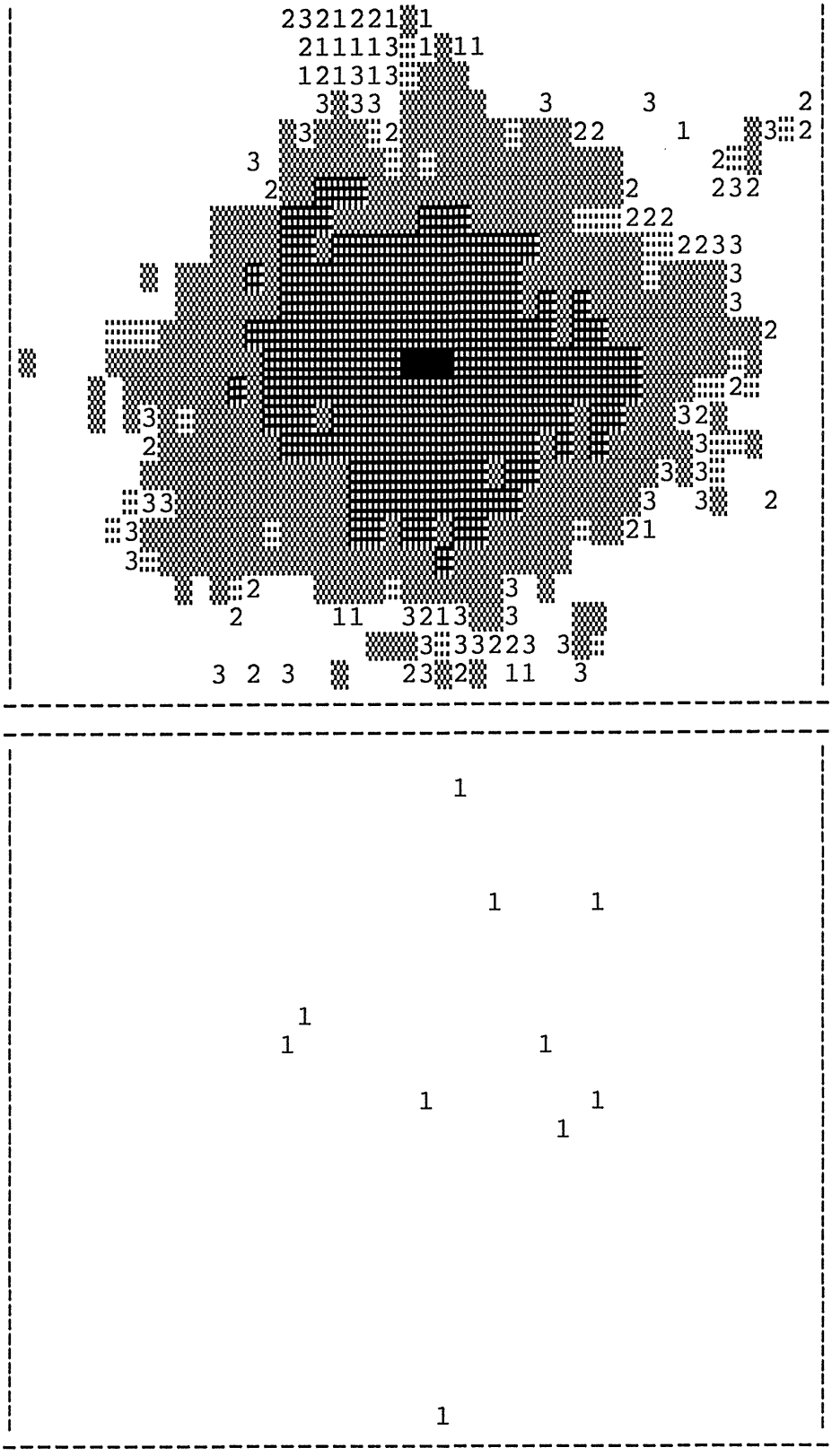


6 okt

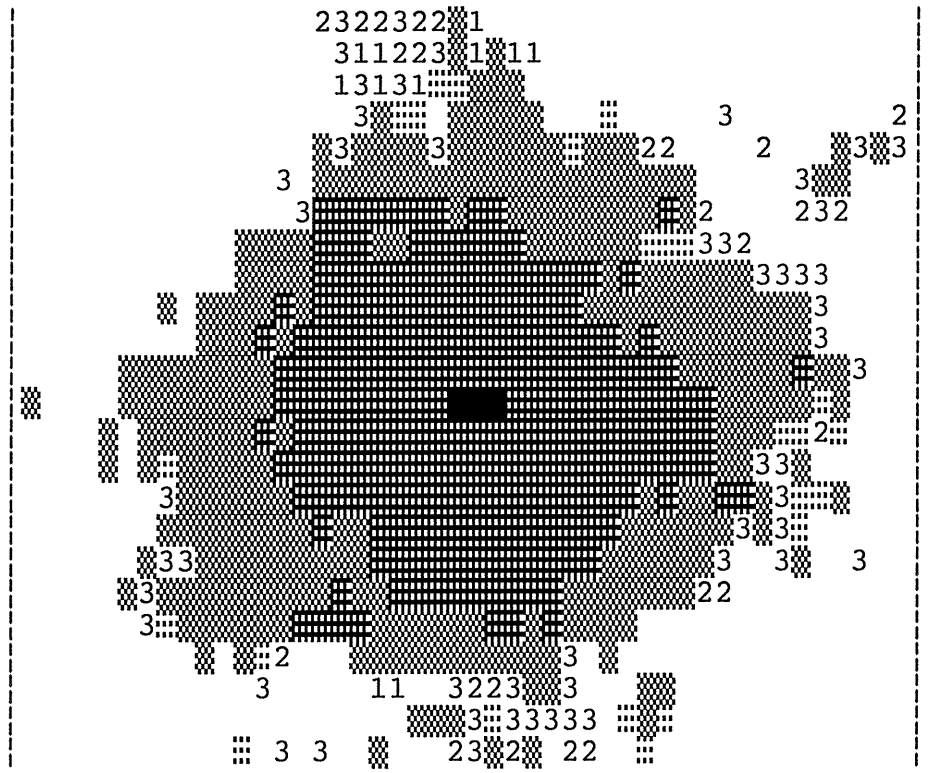




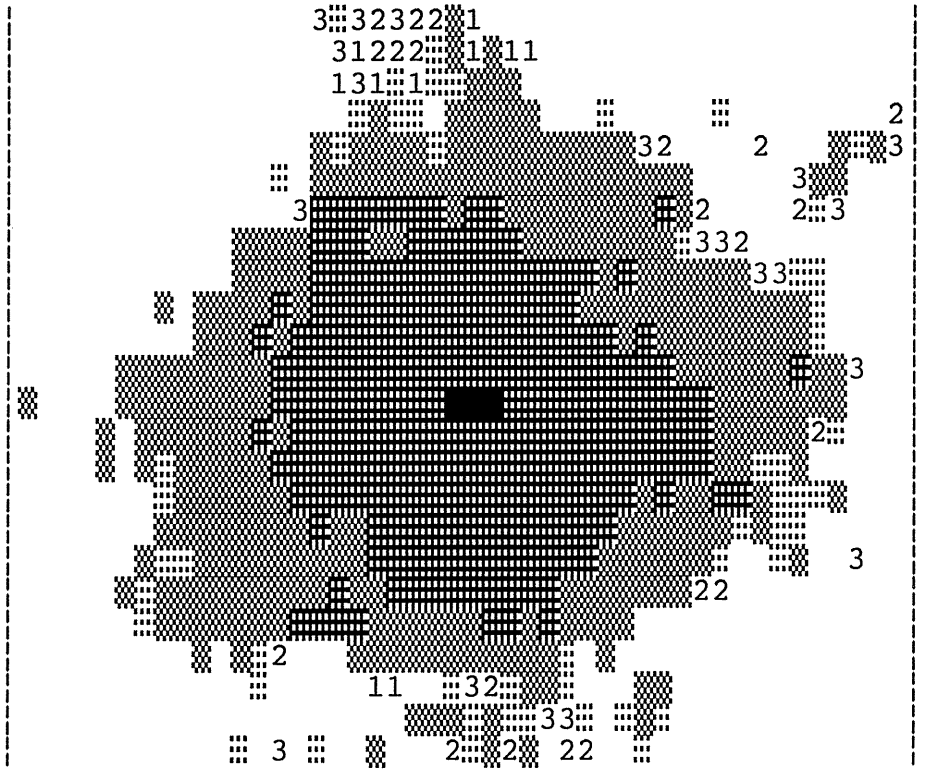
13 okt


2321221 1
211113 1 11
121313
3 33 3 3 2
3 2 22 1 3 2
3 2 2
2 2 232
222
2233
3
3
2
2
3 32
2 3
3 3
33 3 3 2
3 21
3
2 3
2 11 3213 3
3 33223 3
3 2 3 23 2 11 3

20 okt


2322322 1
311223 1 11
13131
3 3 2
3 3 22 2 3 3
3 3
3 2 232
332
3333
3
3
3
2
33
3 3
3 3
33 3 3 3
3 22
3
2 3
3 11 3223 3
3 33333
3 3 23 2 22

27 okt


3 32322 1
31222 1 11
131 1
2
32 2 3
3
3 2 2 3
332
33
3
2
3
22
2
11 32
33
3 2 2 22

Bijlage C

Handleiding voor het gebruik van het simulatieprogramma BYV (verspreiding van BYV door bladluizen in het bietenveld).

In deze bijlage zal achtereenvolgens behandeld worden:
- De files waaruit het simulatiepakket opgebouwd is.
- Beantwoorden van vragen tijdens de dialoog.
- De dialoog.
- Opstarten van het programma met parms.
- Hoe het programma voortijdig te beëindigen.
- Outputfiles.
- Geheugenbeperkingen.
- Zelf parmfiles maken.
- Zelf datafiles veranderen.
- Zelf veranderingen aanbrengen in het programma.

- De files waaruit het simulatiepakket opgebouwd is:

Het pakket omvat dertien files. De namen van de files tussen haakjes zijn niet nodig voor het executeren van het programma. Hieronder volgt een overzicht van de namen van de verschillende files:

- (READDATA.C en READDATA.OBJ)
- (SIMULATE.C en SIMULATE.OBJ)
- BYV.EXE
- BYV.DAT
- TEMP.DAT
- CONSTANT.DAT
- INCUBOLD.DAT
- INCYOUNG.DAT
- LUIZEN0.DAT t/m LUIZEN6.DAT

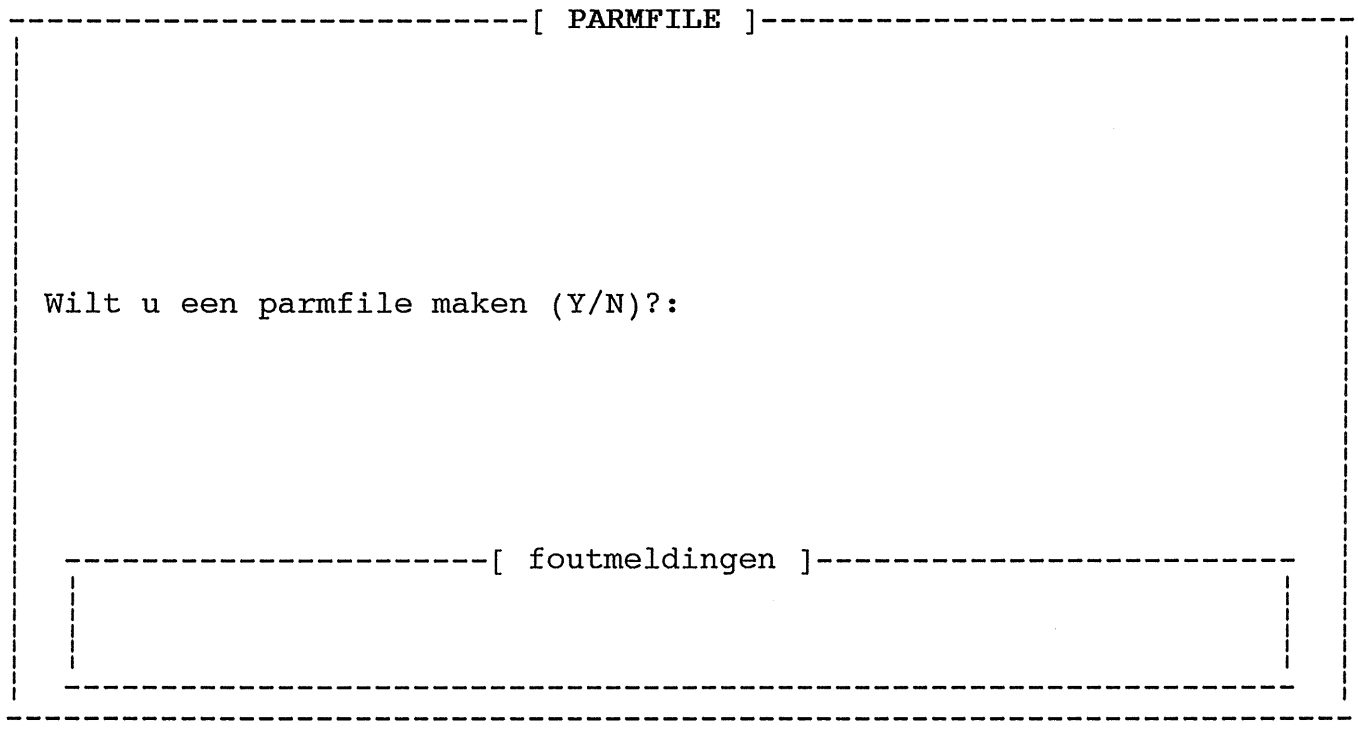
De files luizen0.dat t/m luizen6.dat bevatten gegevens over luizentellingen verricht in het veld. De files incubold.dat en incyoung.dat bevatten gegevens over de incubatietijden van het oude respectievelijk het jonge gewas. (Oud gewas: planten gezaaid op 18 april, jong gewas: planten gezaaid op 20 mei) De file temp.dat bevat temperatuurgegevens over de periode 1-4-86 tot 1-11-86. Voor elke dag uit deze periode zijn acht gegevens opgenomen. De dag is opgesplitst in de volgende acht drieuurssegmenten: 00.00-03.00, 03.00-06.00, 06.00-09.00, 09.00-12.00, 12.00-15.00, 15.00-18.00, 18.00-21.00, 21.00-00.00 uur. In de file constant.dat zijn alle waarden opgenomen welke toebehoren aan de constante variabelen uit het simulatieprogramma. De file byv.dat bevat de opstartwaarden van de laatst uitgevoerde simulatie. De files readdata.c en readdata.obj bevatten de source- en de gecompileerde code van het interactieve deel van het programma. De files simulate.c en simulate.obj bevatten de source- en de gecompileerde code van het simulatie deel van het programma. De twee files met de gecompileerde source code (readdata.obj en simulate.obj) kunnen gelinkt worden tot een executeerbaar programma. Dit is reeds gebeurd. Het executeerbare programma heet byv.exe en kan opgestart worden door het commando byv te geven. Bovengenoemde files moeten alle op de zelfde directory staan om het simulatieprogramma te kunnen gebruiken!

- Beantwoorden van vragen tijdens de dialoog:

Indien het commando byv gegeven is, komt er een dialoog op gang waarin aan de gebruiker diverse vragen gesteld worden. De vragen betreffen de opstartwaarden voor de simulatie. Bij het invoeren van getallen door de gebruiker wordt steeds streng gecontroleerd of de ingevoerde getallen geen waarden hebben welke de toegestane (in het programma vastgelegde) grenzen overschrijden.
Bij het doorlopen van de dialoog zullen, afhankelijk van de beantwoording van de verschillende vragen, verschillende schermen op de monitor verschijnen. Boven in de omrastering van elk scherm staat een vetgedrukt themawoord waaraan gezien kan worden waar de vragen op dat scherm over gaan. Elk scherm heeft andere vragen. Achter de vraag op het scherm, die als eerstvolgende beantwoord moet worden, bevindt zich een cursor. Deze cursor geeft de plaats aan waar het antwoord, dat door de gebruiker ingetoetst moet worden, zal verschijnen. Na het beantwoorden van alle vragen op een scherm zal de vraag, "Moet er nog wat gewijzigd?", boven in het scherm verschijnen. Indien tijdens het beantwoorden van de vragen een fout antwoord ingetoetst is, moet deze vraag bevestigend beantwoord worden. De vragen kunnen dan opnieuw doorlopen worden. Bij een vraag waarachter reeds een antwoord staat, dat niet veranderd hoeft te worden, kan volstaan worden met het geven van een carriage return. Het getal op het scherm zal onveranderd blijven. De cursor springt naar de volgende vraag
Bij de meeste vragen mag slechts een beperkt aantal karakters ingetoetst worden. Is dit aantal bereikt dan zal de cursor, zonder dat er eerst een carriage return gegeven werd, naar de volgende vraag springen. Wordt er toch een carriage return gegeven, dan geldt dit als antwoord op de eerstvolgende vraag. Dit zal wellicht niet de bedoeling geweest zijn. Let bij het beantwoorden van vragen dus altijd goed op of de cursor niet al uit zich zelf naar de volgende vraag gesprongen is.
Onderaan elk scherm is een apart blok opgenomen. Hierin verschijnen foutmeldingen indien de gebruiker een verkeerd karakter ingetoets heeft of als de waarde van het ingetoetste getal buiten de grenzen van het toegestane valt. Verkeerde karakters verschijnen niet op het scherm. Bij het intoetsen van een verkeerd karakter zal de aard van de fout vermeld worden. Bijvoorbeeld indien een cijfer in plaats van een letter ingetoetst werd. Indien een getal met een verkeerde waarde ingetoetst is, zal vermeld worden binnen welke grenzen het getal moet liggen.
Indien het invoeren van een getal nog niet beëindigd is door het geven van een carriage return, kan het getal nog weggehaald worden van het scherm. Hiertoe moet de backspace toets aangeslagen worden. Let op, het gehele getal verdwijnt van het scherm! Alle karakters waaruit het ingevoerde getal bestond, moeten hierna opnieuw ingevoerd worden.

- De dialoog:

De eerste vraag die aan de gebruiker gesteld wordt, betreft het al of niet willen maken van een parmfile. De gegevens die de gebruiker intoetst om een simulatie te draaien, kunnen ook opgeslagen worden op parmfile.

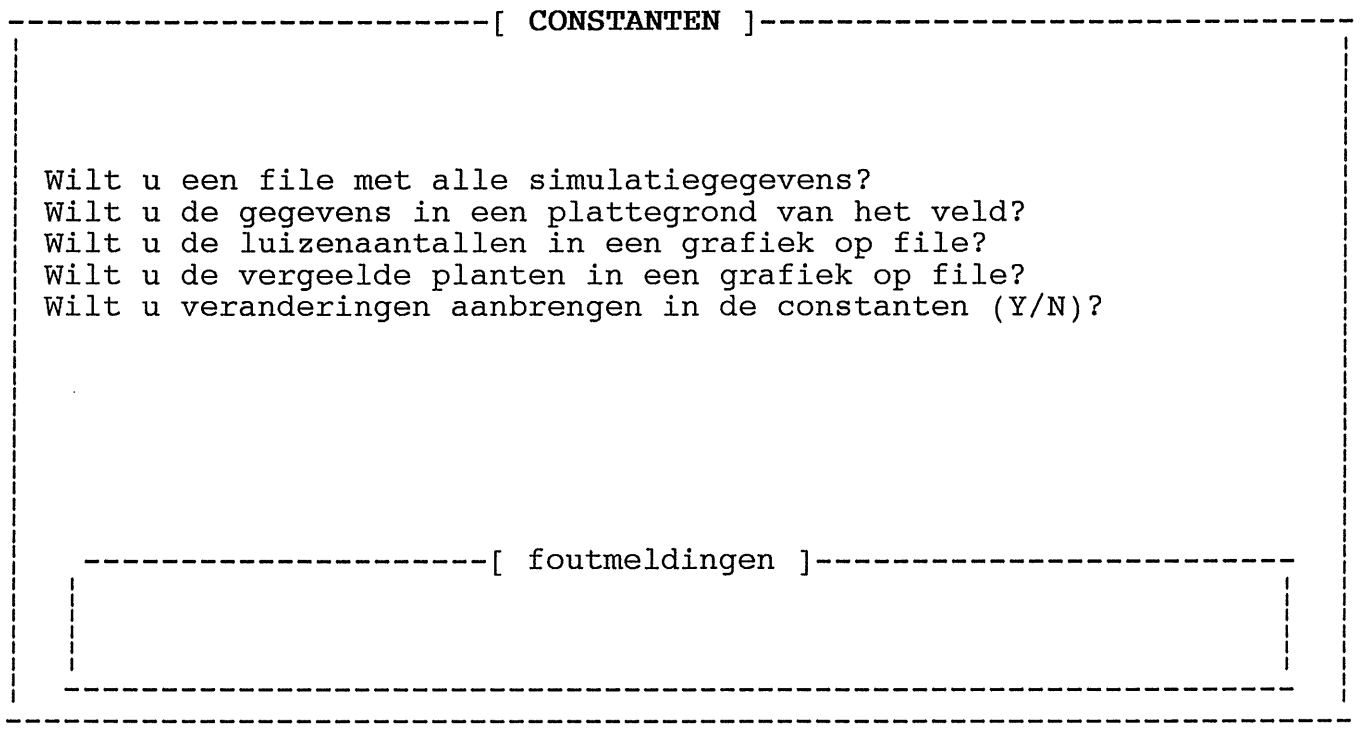


Indien er geen parmfile gewenst is, moet de vraag met N (van no) beantwoord worden. Wordt er gekozen om wel een parmfile te maken, dan verschijnt er een tweede vraag op het scherm. De parmfile die gemaakt zal worden moet door de gebruiker van een naam voorzien worden. Voor het gemak zal in de tekst hieronder voor de naam van de parmfile "naam" gebruikt worden of, als er meerdere (n) parmfiles zijn, de namen: "naam1", "naam2" ... "naamn".

Het scherm ziet er dan als volgt uit:

De naam die hierboven gekozen wordt ("naam"), zal met extentie "par" ("naam.par") op disk opgeslagen worden. Indien de gegevens van deze file nodig zijn voor het opstarten van een simulatie, moet het commando: byv "naam" gegeven worden. Er kunnen ook meerdere parmfiles gemaakt worden. Indien dan het commando: byv "naam1" "naam2" ... "naamn" gegeven wordt, zullen de gegevens uit parmfiles "naam1" t/m "naamn" gebruikt worden om de verschillende simulaties op te starten.

De tweede vraag die aan de gebruiker gesteld wordt, betreft het al of niet willen veranderen van de constanten van het programma.



Er kan gekozen worden uit vier verschillende outputfiles. Voor elke outputfile wordt eerst gevraagd of de gebruiker deze wel wil. Is de betreffende outputfile gewenst, dan moet de vraag met Y van Yes beantwoord worden.
Verder wordt hier de mogelijkheid geboden om de constanten van het programma te veranderen. In de meeste gevallen zal hier de N (van no) ingevuld worden. Constanten zijn niet voor niets constanten en hoeven dan ook meestal niet veranderd te worden. Moet er echter toch een constante van waarde veranderen dan moet op deze vraag met Y (van yes) geantwoord worden. Wordt er op deze vraag een ander antwoord gegeven dan bovengenoemde twee mogelijkheden, dan zal dit een foutmelding tot gevolg hebben.

Indien op de vraag, over het veranderen van de constanten van het programma, bevestigend geantwoord is, zal de volgende schermopmaak verschijnen:

```
--------------------[ CONSTANTEN VERANDEREN ]-------------------------
|                                                                    |
|  Maximum leeftijd van de luizen (dagen): 42                        |
|  Retentie periode van het virus (uren): 12                         |
|  Zuigtijd om het virus op te nemen (uren): 2                       |
|  Zuigtijd om het virus af te geven (uren): 1                       |
|  Minimum vliegtemperatuur (graden C): 15                           |
|  Maximum vliegtemperatuur (graden C): 30                           |
|  Minimum looptemperatuur (graden C): 10                            |
|  Optimum looptemperatuur (graden C): 30                            |
|  Maximum looptemperatuur (graden C): 50                            |
|  Minimum temperatuur voor bladverschijning (graden C): 1           |
|  Minimum temperatuur voor bladgroei (graden C): 3                  |
|  Optimum temperatuur voor bladgroei (graden C): 24                 |
|  Maximum temperatuur voor bladgroei (graden C): 48                 |
|  Leeftijd waarbij maximale loopsnelheid (uren): 600                |
|  Maximale loopsnelheid van de luis (cm/uur): 125                   |
|    ----------------------[ foutmeldingen ]-----------------------  |
|    |                                                            |  |
|    |                                                            |  |
|    |                                                            |  |
|    ------------------------------------------------------------  |
|                                                                    |
----------------------------------------------------------------------
```

De getallen die ingevuld worden voor de diverse constanten worden niet op waarde gecontroleerd. De waarde die toegekend wordt aan een constante mag zo groot en zo klein zijn als de gebruiker wenst. De waarde moet echter wel geheel en positief zijn. Dit geldt voor alle constanten. Hier wordt op gecontroleerd bij het invoeren van de getallen.
Indien alle vragen beantwoord zijn, komt boven in het scherm de vraag; Moet er nog wat gewijzigd?. Deze vraag moet bevestigend beantwoord worden indien er bij het beantwoorden van de vragen een fout gemaakt is. Er wordt dan opnieuw gelegenheid gegeven om de constanten van waarde te veranderen. Wordt er tijdens het intoetsen van een antwoord een fout gemaakt, er is dan nog geen carriage return gegeven, dan kan de backspace toets aangeslagen worden. De waarde voor de betreffende constante kan nu opnieuw ingetoetst worden.
Achter de vragen zijn reeds waarden voor de constanten ingevuld. Bij waarden die niet veranderd hoeven te worden, kan volstaan worden met het geven van een carriage return. De waarde die op het scherm staat, zal dan onveranderd blijven.

Nadat de gebruiker de constanten de gewenste waarden gegeven heeft en de vraag: Moet er nog wat gewijzigd?, dus ontkennend heeft beantwoord, komt de volgende schermopmaak op het beeldscherm:

```
----------------------[ HET BIETENVELD ]----------------------------
|        Dit is een simulatie van de verspreiding van BYV door      |
|                    bladluizen in een bietenveld.                  |
|                                                                   |
| Om de hoeveel dagen wilt u een print van de simulatie hebben?: 7  |
| Geef een beginwaarde (zaadje) voor de uniforme verdeling:         |
|                                                                   |
| De rijen van het bietenveld liggen evenwijdig aan de x_as.        |
| De lengte en de breedte van het bietenveld moeten in gehele       |
| meters opgegeven worden.                                          |
| De lengte van het veld correspondeert met de x_as.                |
| De breedte van het veld correspondeert met de y_as.               |
| Wat is de lengte van het bietenveld (x_as)?: 12                   |
| Wat is de breedte van het bietenveld (y_as)?: 12                  |
| Hoeveel planten staan er per meter in de lengte (x_as)?: 3.90000  |
| Hoeveel planten staan er per meter in de breedte (y_as)?: 2.00000 |
|                                                                   |
|  ----------------------[ foutmeldingen ]-------------------------|
|  |                                                              |  |
|  |                                                              |  |
|  |                                                              |  |
|  ---------------------------------------------------------------  |
-------------------------------------------------------------------
```

De eerste vraag moet beantwoord worden met een getal groter dan nul. Wordt er een waarde gegeven groter dan het aantal te simuleren dagen, dan wordt er slechts één plaatje geprint. Die van de beginsituatie van de simulatie. Normaliter staat er reeds een getal op het scherm. Dit getal is de waarde welke is gebruikt bij de vorige simulatie. Is als getal een 7 ingevuld, dan zal om de week een print gemaakt worden van de simulatie. Duurt de simulatie 56 dagen (8 weken) dan worden er negen plaatjes gemaakt. Eén plaatje van de begintoestand van de simulatie en vervolgens één voor iedere gesimuleerde week. Alle plaatjes en andere gegevens die gedurende het simuleren weggeschreven worden, komen allemaal in speciaal daarvoor geopende outputfiles te staan. Hoe deze outputfiles er uit zien, wordt in een ander onderdeel van dit hoofdstuk beschreven.

Het zaadje dat gevraagd wordt, moet een getal zijn groter dan nul. Het maximale aantal cijfers dat hier ingetoetst kan worden is drie. Hiermee ligt de hoogst mogelijke waarde voor het zaadje vast op 999. Het is raadzaam om simulaties meerdere keren te draaien met verschillende zaadjes. Het is mogelijk dat een ander zaadje iets andere simulatieresultaten tot gevolg heeft.

Vervolgens moeten de maten van het veld nog ingelezen worden. De minimale dichtheid van planten is één plant per vierkante meter. Wordt er dus een waarde voor het aantal planten per meter in de lengte of in de breedte gegeven kleiner dan 1, dan volgt een foutmelding. Bij een dichtheid van 1 plant per vierkante meter kan het veld maximaal 100 meter lang en 50 meter breed zijn. Het maximale aantal planten in het

veld is namelijk 5000. Bij hogere dichtheden horen dus kleinere lengten en breedtes. Het programma controleert of er wel aan bovenstaande eisen voldaan wordt. Het aantal planten per meter in de lengte en in de breedte hoeven geen gehele aantallen te zijn. Voor de lengte en de breedte van het veld mogen maximaal twee cijfers ingetypt worden. Is dit maximum bereikt dan springt de cursor vanzelf naar de volgende vraag. Er hoeft dus geen carriage return gegeven te worden. Wordt deze carriage return toch gegeven dan geldt ze als antwoord op de volgende vraag. Indien een vraag slechts met een carriage return beantwoord wordt, wordt de waarde die achter deze vraag staat als opstartwaarde genomen.

Ook bij dit scherm wordt gevraagd of er nog wat veranderd moet worden na beantwoording van alle vragen. Is dit niet het geval dan moet de vraag met N (No) beantwoord worden. Zijn er bij het typen vragen fout beantwoord dan moet er Y (Yes) geantwoord worden. Hierdoor zal er gelegenheid gegeven worden om de vragen opnieuw te beantwoorden. Bij vragen die goed beantwoord waren, kan volstaan worden met het geven van een carriage return.

Indien de maten van het veld ingelezen zijn, verschijnt het volgende scherm:

```
-------------------------[ ALGEMEEN ]--------------------------------
|                                                                   |
|                                                                   |
|  Uitzetten bladluizen - infecteren planten:                       |
|  Wat is de zaaidatum van de bietenplanten?                        |
|  maand (in cijfers): 5                                            |
|  dag: 20                                                          |
|                                                                   |
|  Wat is de startdatum van de simulatie?                           |
|  maand (in cijfers): 6                                            |
|  dag: 25                                                          |
|                                                                   |
|  Wilt u een of meerdere planten infecteren (Y/N)?:                |
|  Wilt u zelf luizen uitzetten (Y/N)?:                             |
|                                                                   |
|  Hoe lang moet de simulatie duren (in dagen)? 127                 |
|                                                                   |
|   ----------------------[ foutmeldingen ]-------------------------|
|   |                                                               |
|   |                                                               |
|   |                                                               |
|   ---------------------------------------------------------------|
|                                                                   |
-------------------------------------------------------------------
```

Indien de luizen niet op het moment van infectie van de planten in het veld moeten komen, moet op de eerste vraag het aantal dagen, dat ligt tussen de infectie en de introductie van de luizen, opgegeven worden. Aangezien de periode waarbinnen gesimuleerd kan worden, ligt tussen 1 april 1986 en 1 november 1986, zijn de waarden die voor de zaaidatum en de startdatum in aanmerking komen beperkt. De zaaidatum van het bietengewas moet liggen voor de datum waarop de simulatie begint. Dat de ingevoerde gegevens aan bovengenoemde voorwaarden voldoen, wordt door het programma gecontroleerd. Zijn beide data ingevoerd, dan wordt gevraagd of ze nog gewijzigd moeten worden. Dit wijzigen gaat gelijk zoals boven bij de andere schermen is beschreven. Is op de vraag of er nog iets gewijzigd moet worden ontkennend geantwoord, dan verschijnt vervolgens knipperend op het scherm, "Even geduld a.u.b!". In deze tijd wordt berekend hoe veel de planten groeien voor dat de eigenlijke simulatie begint. Zijn de berekenigen hiertoe beëindigd, dan komt het oude scherm weer terug. De volgende twee vragen veroorzaken het verschijnen van nieuwe schermen indien ze met Y (Yes) beantwoord worden. Indien de vraag over het infecteren van planten met N (No) beantwoord wordt, springt de cursor naar de volgende vraag. Er worden dan simpelweg geen planten met het virus besmet. In het geval dat de gebruiker voor Y (Yes) gekozen heeft, verschijnt er een scherm met vragen over welke plant er met het virus besmet moet worden en of er daarnaast nog meer planten besmet moeten worden. Indien de vraag of de gebruiker zelf luizen wil uitzetten met N (No) beantwoord wordt, verschijnt er een nieuw scherm. Hier kan de gebruiker kiezen uit een aantal luizenverdelingen zoals die in het veld waargenomen zijn. Werd deze vraag echter met Y (Yes) beantwoord, dan verschijnt een scherm waarop vragen staan voor het zelf uitzetten van luizen. Hoe al deze

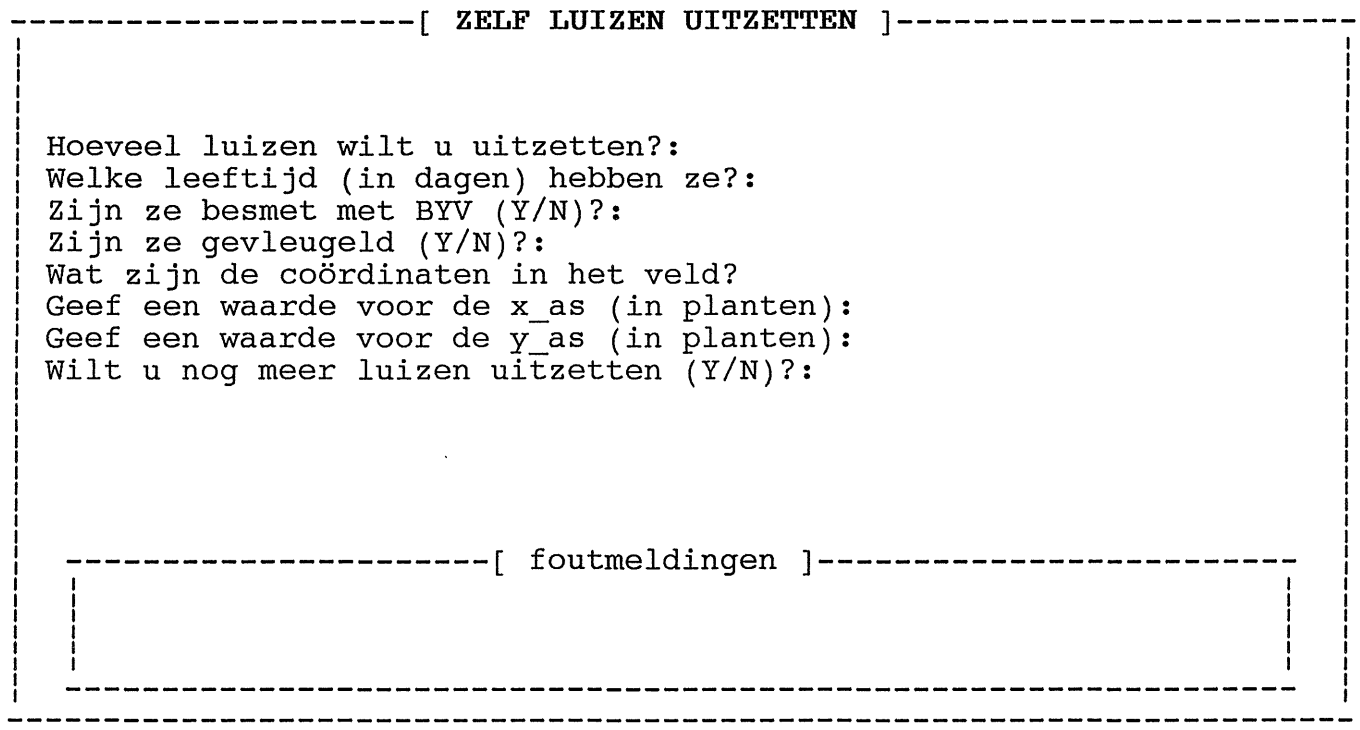
schermen er uit zien zal hieronder beschreven worden. Voor het beantwoorden van de laatste vraag over de lengte van de te simuleren periode moet, voordat de carriage return toets aangeslagen wordt, goed gekeken worden of het ingetoetste getal de juiste waarde heeft. Is de carriage return toests aangeslagen dan begint het simulatieprogramma te lopen. Het programma controleert wel eerst nog of de periode waarover de simulatie moet lopen niet te lang is waardoor buiten de toegestane periode van 1-4-86 tot 1-11-86 gesimuleerd zou moeten worden.

Indien op de vraag; Wilt u een of meerdere planten infecteren (Y/N)?:, bevestigend geantwoord is, verschijnt het volgende scherm:

```
----------------------[ PLANTEN INFECTEREN ]--------------------------
|                                                                     |
|                                                                     |
|                                                                     |
|                                                                     |
|  Wat zijn de coördinaten in het veld?                               |
|  Geef een waarde voor de x_as (in planten):                         |
|  Geef een waarde voor de y_as (in planten):                         |
|  Wilt u nog meer planten infecteren (Y/N)?:                         |
|                                                                     |
|                                                                     |
|                                                                     |
|                                                                     |
|    ----------------------[ foutmeldingen ]-------------------------  |
|    |                                                             |  |
|    |                                                             |  |
|    |                                                             |  |
|    -------------------------------------------------------------  |
-----------------------------------------------------------------------
```

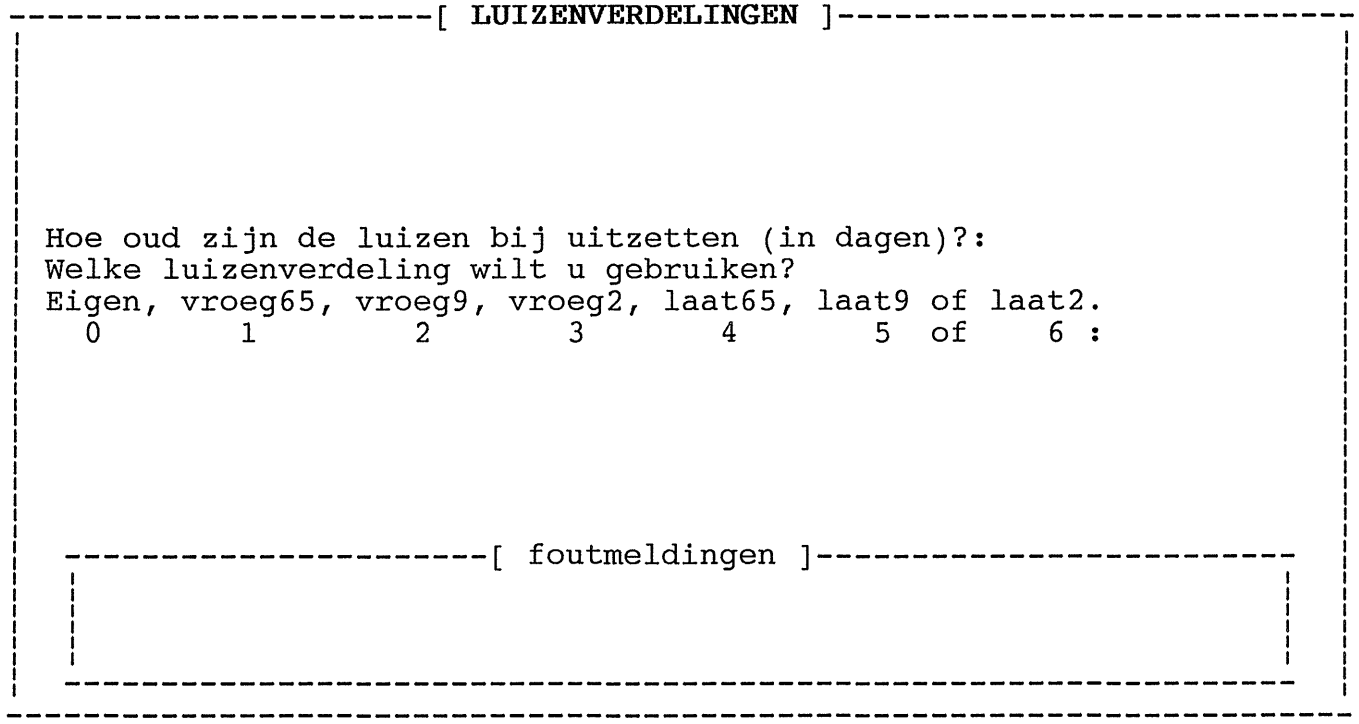
Om de waarden voor de x_as en de y_as in planten te kunnen geven, moet het volgende voor ogen gehouden worden: De planten in de simulatie zijn in een assenstelsel geplaatst. Alle planten staan in het eerste kwadrant van dit assenstelsel. Op de assen zelf staan ook planten. Op plaatsen waar de lijnen; x = 0, x = 1, x = 2 enz. en de lijnen; y = 0, y = 1, y = 2 enz. elkaar kruisen, staan planten. De plant links onder in het veld heeft dus de coördinaten (0,0). Het aantal planten in de breedte en in de lengte van het veld hangt af van de door de gebruiker opgegeven waarden. De rijen van het bietenveld liggen evenwijdig aan de x_as! Moeten er meerdere planten geïnfecteerd worden, dan moet de vraag; Wilt u nog meer planten infecteren (Y/N)?:, met Y (Yes) beantwoord worden. Voordat deze vraag echter gesteld wordt, wordt nog gevraagd of de ingevoerde gegevens nog veranderd moeten worden. Hoe deze vraag afgehandeld moet worden, is al eerder beschreven.

Indien de gebruiker zelf luizen wil uitzetten in het bietenveld, moet op de volgende vragen antwoord gegeven worden:



De vragen in dit scherm spreken voor zich. Er worden groepjes luizen uitgezet die allemaal dezelfde eigenschappen krijgen. de grootte van de groepjes bepaald de gebruiker zelf. Er mogen in totaal niet meer dan 900 luizen uitgezet worden. De leeftijd van een luis mag niet hoger zijn dan 42 dagen. Op de vragen waar (Y/N) achter staat, moet met een Y van Yes of een N van No geantwoord worden. De coördinaten, van de plant waar de luizen op uitgezet moeten worden, worden op dezelfde manier bepaald als beschreven bij het infecteren van planten. Ook bij dit scherm zal de vraag gesteld worden of er nog fouten weggewerkt moeten worden voordat de laatste vraag, over het al of niet uitzetten van nog meer luizen, afgehandeld wordt.

Indien de gebruiker niet wenst om zelf luizen uit te zetten, maar een luizenverdeling wil gebruiken welke van te voren vast ligt, dan moet de vraag uit onderstaande scherm beantwoord worden.



De luizenverdelingen die hierboven gegeven zijn, hebben allemaal betrekking op de veldproeven die uitgevoerd zijn door Corné Kempenaar. De files luizen0.dat t/m luizen6.dat bevatten de getallen van luizentellingen verricht in diverse veldproeven bij de Minderhoudhoeve. De getallen in de namen van de files komen overeen met de getallen die in het scherm hierboven gekozen kunnen worden. Ergo, wordt getal 3 gekozen dan wordt file luizen3.dat geopend. Van deze file worden vervolgens de gegevens van de luizenaantallen in het veld gelezen.

- Opstarten van het programma met parms:

Het simulatieprogramma opgestart met byv, doorloopt eerst een dialoog met de gebruiker. Het simulatieprogramma kan echter ook opgestart worden op een andere manier. Achter het commando byv moeten dan één of meerdere filenamen gegeven worden. Deze filenamen (parms) moeten de namen van parmfiles zijn. De parmfiles staan op schijf met de extentie ".par". Bij het gebruik van de namen van deze parmfiles achter het commando byv, moet deze extentie weggelaten worden. Een parmfile is een file waarin alle informatie staat welke nodig is voor het opstarten van één simulatie. Deze gegevens die anders tijdens de dialoog met de gebruiker interactief ingelezen zouden worden, worden nu van de parmfile ingelezen. Worden achter het commando byv 10 parms gegeven dan zullen er 10 simulaties gedaan worden. De outputfiles zullen dan genummerd zijn van 1 tot 10 (zie bij: Outputfiles).

- Hoe het programma voortijdig te beëindigen:

Indien de gebruiker het programma wil beëindigen voordat de simulatie /simulaties klaar is/zijn, dan kan dat op verschillende manieren. Om er zeker van te zijn dat de informatie die tot dan toe naar file geschreven is niet verloren gaat, moet het programma beëindigd worden met ^C (control (ctrl) C). Zodra er informatie naar het beeldscherm geschreven wordt vanuit het programma, zal het programma onderbroken worden. Alle in het programma geopende files worden netjes gesloten en de informatie die tot dan toe naar file geschreven is, is niet verloren. Het programma kan ook onderbroken worden met Ctrl Alt Del. Met deze toetsencombinatie wordt de opdracht voor een system reset gegeven. Geopende files worden nu niet gesloten. Informatie die reeds naar schijf geschreven was, gaat hierdoor verloren. Het uit en weer aan zetten van de computer beëindigd het programma eveneens maar met dezelfde gevolgen voor de naar schijf geschreven gegevens als bij het commando Ctrl Alt Del. Het commando Ctrl Break heeft het zelfde resultaat als het commando Ctrl C.

- Outputfiles:

Er kunnen vier verschillende outputfiles op schijf gezet worden vanuit het programma. Bij het opstarten van iedere simulatie moet er gekozen worden welke outputfiles gewenst zijn. Alle combinaties zijn hierbij toegestaan. De verschillende outputfiles zijn te herkennen aan de extentie die er aan meegegeven wordt. Alle outputfiles beginnen met "sim". Na "sim" volgt nog een getal (#). Dit getal is 1 indien er één simulatie gedaan is. Het getal kan hoger zijn indien het programma opgestart is met meerdere parmfiles achter het commando byv (zie bij: Opstarten van het programma met parms). Elke keer als het commando byv opnieuw gebruikt wordt, wordt weer opnieuw begonnen met het nummeren van de outputfiles. Oude files worden hierbij overschreven. Een outputfile kan de volgende vier namen hebben, sim#.dat, sim#.pic, sim#.pla en sim#.lui.

- sim#.dat

In deze file worden algemene gegevens weggeschreven. De opstartwaarden en de waarden van de constanten aan het begin. Verder wordt, om de zoveel tijd (afhankelijk van het opgegeven tijdsinterval), vermeld, wat de datum en de tijd zijn, hoe warm het is (temperatuur in °C), het

totale aantal bladluizen, het aantal besmette bladluizen, het aantal gevleugelde bladluizen, het aantal bladeren per plant, het totale bladoppervlak van de bietenplanten, het aantal luisdagen getelt vanaf het begin van de simulatie, de predatiedruk en de leeftijdsverdeling van de luizen. De leeftijdsverdeling van de luizen is weergegeven in zes getallen, ":: L1 L2 L3 L4 adult1 adult2 ::". Er zijn vier larvale stadia, L staat voor larvaal stadium. Adult1-luizen zijn luizen jonger dan 360 uur en ouder dan 144 uur. Adult2-luizen zijn luizen jonger dan 1008 uur en ouder dan 360 uur.

- sim#.pic

In deze file worden gegevens van de planten en van de luizenaantallen in een plattegrond van het veld weggeschreven. De eerste plattegrond bevat informatie over de planten, de volgende plattegrond bevat informatie over het aantal luizen. Zijn er in het veld 48 * 24 planten aanwezig, dan zal de plattegrond 48 karakters breed en 24 karakters lang zijn. Er kunnen verschillende karakters gebruikt zijn.

In de plattegrond voor de planten:
-een spatie voor een gezonde plant,
-een cijfer voor een plant met besmette bladeren. Het cijfer staat voor de klasse van de plant. De klasse is een indicatie voor het percentage besmet oppervlak. Is de klasse van de plant 10, dan wordt er een T weggeschreven,
-een rastertje voor een vergeelde plant. Er zijn vier rastertjes, hoe "zwarter" het rastertje hoe groter het percentage besmet blad.

In de plattegrond voor de luizenaantallen:
-een spatie waar zich geen luizen bevinden,
-een cijfer dat aangeeft hoeveel luizen er op die plant zitten,
-een rastertje indien er meer dan 9 luizen op een plant zitten.

- sim#.pla

In deze file wordt een grafiek weggeschreven van het aantal vergeelde planten tegen de datum. Om de zoveel tijd (afhankelijk van de hiervoor opgegeven periode), wordt de volgende informatie in de grafiek geschreven, *(#) C#. Het sterretje is een punt van de grafiek. Het eerste getal (tussen haakjes) staat voor het aantal vergeelde planten. Het tweede getal (achter de C) staat voor het aantal besmette planten (Contaminated).

- sim#.lui

In deze file wordt een grafiek weggeschreven van het totale aantal luizen in de simulatie tegen de datum. Om de 24 uur wordt de volgende informatie in de grafiek geschreven, *(#) S# V#. Het sterretje is ook hier een punt van de grafiek. Het eerste getal (tussen haakjes) staat voor het totale aantal luizen in de grafiek. Het tweede getal (na de S van Simulatie) geeft aan hoeveel luizen er op de middelste 39 planten in de simulatie getelt zijn. Het laatste getal (na de V van Veld) geeft aan hoeveel luizen er op dezelfde 39 planten in het veld getelt zijn.

- Geheugenbeperkingen:

De omvang van het geheugen dat door de gebruikte C compiler gebruikt kan worden is 128K, een 64K programmasegment en een 64K datasegment. Hiermee valt deze compiler in de catagorie kleine compilers. Deze

geheugenbeperking stelt grenzen aan het aantal planten en luizen dat aangemaakt kan worden tijdens een simulatie. Het geheugen van de Bondwell AT's, waar de simulaties op gedraaid zijn, is 520 Kbyte. Met een "grote" C compiler zouden er aanzienlijk meer planten en luizen aangemaakt kunnen worden tijdens een simulatie. Het maximale aantal luizen dat aangemaakt kan worden is nu ruim 900 (bij 5000 planten). Luizenaantallen groter dan 900 zijn daarom in het model niet mogelijk.

- Zelf parmfiles maken:

Het programma biedt de mogelijkheid om de gegevens die tijdens de dialoog ingevoerd worden, op te slaan in een parmfile. Deze gegevens kunnen natuurlijk ook met een gewone line editor op file gezet worden. De kans dat daarbij fouten gemaakt worden is echter groot. De gegevens moeten allemaal aan bepaalde voorwaarden voldoen. Wordt de parmfile gemaakt met behulp van het programma, dan controleert het programma de gegevens op correctheid voordat ze naar file geschreven worden. Een simulatie die opgestart wordt met een parmfile die met het programma gemaakt is, gaat nooit fout. Bij simulaties opgestart met parmfiles met foute waarden, kunnen er onvoorspelbare dingen gebeuren!

- Zelf datafiles veranderen:

Het simulatieprogramma leest van verschillende datafiles gegevens in voor de simulaties. Deze files kunnen door de gebruiker aangepast worden. Wel moet er dan op gelet worden dat dit op de juiste wijze gebeurd. Van alle datafiles zal hieronder vermeld worden volgens welke "format" de gegevens op file gezet moeten worden.
- temp.dat: "___". Vb.: "-9_", "11_", "_5_" enz.
- incubold.dat en incyoung.dat: "___". Vb.: "__6", "_35" enz.
- luizen#.dat: "____". Vb.: "___9", "_296" enz.
- constant.dat: Elke waarde op een nieuwe regel, geen "format".
- byv.dat: Elke waarde op een nieuwe regel, geen "format".

- Zelf veranderingen aanbrengen in het programma:

Om veranderingen aan te brengen in het programma moet de source code veranderd worden met een line editor. Het programma is geschreven in de programmeertaal C en voldoet aan de spellingsregels zoals die beschreven zijn in het boek, "De programmeertaal C" van ir.L.Ammeraal. De source code is over twee files verdeeld, readdata.c en simulate.c (zie ook bijlage B: Het C programma). Na verandering van de source code moet de file waarin de wijzigingen aangebracht zijn, gecompileerd worden met een C-compiler. Deze compilatie levert een object file op. Deze object file moet samen met de object file van het andere deel van het programma, gelinked worden. Na linken van beide object files is er weer een file met executeerbare code (byv.exe).

Bijlage C. Het C programma

- De opzet van het programma:

Vanwege de omvang van het programma is het in twee delen opgesplitst. Het ene deel bevat alle functies nodig voor de dialoog met de gebruiker en het initialiseren van alle globale variabelen. Dit is het interactieve deel. Het andere deel bevat alle functies voor de uiteindelijke simulatie alsmede het hoofdprogramma. Dit is het simulatie deel.
Van het interactieve deel wordt geen volledige verklaring gegeven. Belangrijker is hoe het gebruikt moet worden. Dit wordt uitgebreid omschreven in bijlage A, "Handleiding voor het gebruik van het simulatieprogramma BYV".
Van het simulatie deel van het programma, wordt in deze paragraaf een uitgebreide verklaring per functie gegeven.
Elke luis en elke bieteplant wordt vertegenwoordigd door een plaats in de geheugenruimte van de computer. Voor elke bladluis wordt in een structuur een aantal gegevens vastgelegd. Een structuur in C is te vergelijken met een record in PASCAL. Een structuur bestaat uit velden. De gegevens van de bladluizen liggen opgeborgen in velden van een structuur. De structuur voor de luis neemt 184 geheugenbits in beslag.
Voor elke plant wordt ook een aantal gegevens vastgelegd. Ook deze gegevens staan bij elkaar in een structuur. Voor elke plant neemt dit een geheugenruimte van 72 bits in beslag.
Elke structuur heeft een adres, een geheugenadres. Een variabele die dit adres bevat, wordt een pointer genoemd. Voor de planten is er een matrix van pointers gedeclareerd. Voor elke bieteplant in het te simuleren veld is zoals reeds gezegd een structuur gedeclareerd. Elke pointer uit de matrix wijst naar de structuur van een plant. De coördinaten van de plant komen overeen met de coördinaten van de pointer in de matrix. De matrix is 100 bij 50 elementen groot. Deze matrix neemt daardoor een geheugenruimte van 5000 * 48 bits in beslag. De afmetingen van deze matrix vormen de beperking voor het aantal planten in het veld bij simulatie op een PC/AT. De x en de y coördinaten van de luis zijn vastgelegd in centimeters. Om te berekenen met welke plantcoördinaten deze luiscoördinaten corresponderen, moeten de luiscoördinaten gedeeld worden door het aantal centimeters tussen de planten onderling. De deling is een integer deling zodat de uitkomst een geheel getal zal zijn. De berekeningen voor de x en de y coördinaat moeten afzonderlijk geschieden daar de onderlinge afstanden tussen de planten in de x en de y richting kunnen verschillen.
De luizen zijn ondergebracht in een double linked list. Dit wil zeggen dat in de structuur van elke luis twee pointers opgenomen zijn. Eén die wijst naar het adres van een luis die voor en één die wijst naar het adres van een luis die er na komt. Ook zijn er twee pointers gedeclareerd waarvan de ene wijst naar het adres van de eerste luis in de double linked list en de andere naar het adres van de laatste luis. Wijzen beide pointers naar het zelfde adres dan bestaat de double linked list slechts uit één luis. Wijzen beide pointers naar NULL (een leeg adres) dan is de double linked list leeg.

- De globale variabelen van het programma:

De globale variabelen van een programma zijn variabelen die vanaf het punt van declaratie niet alleen in het hoofdprogramma maar ook in de functies onder het punt van declaratie gebruikt mogen worden. In C mogen, anders dan in een taal als PASCAL, globale variabelen overal tussen de functies gedeclareerd worden. Indien de globale variabelen aan het begin van de file gedeclareerd worden, dan mogen de variabelen in alle functies gebruikt worden.
De structuur voor een luis heet struct aphid en de structuur voor een bieteplant heet struct beet. De pointer naar de eerste luis in de double linked list heet first_aphid, die naar de laatste last_aphid. De pointers naar luizen, gedeclareerd binnen de struct aphid zelf heten next en pred (respectievelijk de volgende en de vorige). Bij het doorlopen van de double linked list van luizen wijst this_aphid achtereenvolgens naar de verschillende luizen in de double linked list. De matrix van pointers, waarvan de pointers wijzen naar de structuren van de bieteplanten, heet field.

- De struct aphid bevat de volgende gegevens:
- Tijdstip van geboorte van de luis: time_of_birth.
- Tijdstip van besmetting van de luis: time_of_contamination.
- Tijdstip waarop de luis van plek wisselt: time_of_switch.
- x_ en y_coördinaten van de luis: x, y.
- Boolean voor gevleugeld zijn van de luis: winged.
- Twee pointers naar een struct aphid: next, pred.

- De struct beet bevat de volgende gegevens:
- Tijdstip van infectie van de plant: time_of_contamination.
- Klasse van besmetting van de plant: class.
- Het aantal luizen op de plant: nbr_of_aphids.
- Het aantal gezonde bladeren aan de plant: healthy_leaves.
- Boolean voor vergeeld zijn van de plant: yellowed.

Gedurende de simulatie zullen de gegevens uit struct beet en struct aphid telkens nodig zijn. De gegevens van de luis worden bereikt door voor de naam van het betreffende veld this_aphid-> te plaatsen. Voor een bieteplant moet field[x][y]-> voor de naam van het veld geplaatst worden. X en y zijn hier de coördinaten van de plant in het veld. Vb. Het bijstellen van het tijdstip van acquisitie van een luis gaat zo: this_aphid->time_of_contamination = time. Time is een globale variabele welke het actuele tijdstip van de simulatie aangeeft.
- De #define's:

- Omdat in C geen kwadrateringsfunctie bestaat, wordt dit met behulp van een define gedefinieerd:
  #define POWER(x) (x)*(x)
  Dankzij deze definitie kan in het programma het volgende voorkomen:
  i = POWER(5+2); waarna i gelijk zal zijn aan 49.

- Omdat in het programma een aantal constanten onderscheiden worden die we ondanks het feit dat het constanten zijn wel willen kunnen veranderen, wordt met behulp van define een nieuw type variabele gedefinieerd:
  #define CONSTANT int
  Door nu te schrijven de declaratie: CONSTANT MAX_AGE; weten we dat MAX_AGE een constante is. MAX_AGE is nu een variabele van het type integer. Natuurlijk zou de declaratie: int MAX_AGE; hetzelfde resultaat opgeleverd hebben. Minder duidelijk dan is echter dat we MAX_AGE als constante willen beschouwen. CONSTANT gedeclareerde variabelen worden in het simulatie deel van het programma nooit veranderd. Alleen in het deel van het programma waar de dialoog met de gebruiker plaatsvindt, kunnen constanten van waarde veranderd worden.
- De maximale lengte van het bietenveld is door middel van:
  #define MAXLENGTH 100
  onveranderlijk vastgelegd op 100 planten. De rijen van een bietenveld liggen evenwijdig aan wat in het simulatieprogramma de lengte van het veld genoemd wordt.
- De maximale breedte van het bietenveld is door middel van:
  #define MAXWIDTH 50
  onveranderlijk vastgelegd op 50 planten. De breedte van het bietenveld ligt dwars op de rijen van een "echt" veld.
- Het aantal temperatuurgegevens dat zich in de file temp.dat bevindt, bedraagt 1712. Na:
  #define TEMP_GEG 1713
  zal de declaratie int temp[TEMP_GEG]; een array van 1713 elementen opleveren met de elementen temp[0] t/m temp[1712]. Het element temp[0] wordt niet gebruikt.
- Het maximaal voorkomende aantal bladeren per plant wordt door:
  #define MAX_LEAVES 80
  met een veilige marge op 80 gesteld.
- Het maximale aantal luizen dat door de gebruiker uitgezet mag worden bedraagt 900 stuks. Vandaar:
  #define MAX_APHIDS 900
- Het maximale aantal karakters dat door de functie scan_input() ingelezen kan worden is 8. Vandaar:
  #define MAX_CHAR 8
- Er is een lege string gedefinieerd door middel van:
  #define EMPTY_STRING " (70 spaties) "
  Wordt er nu een print opdracht gegeven met een EMPTY_STRING dan zal een lege regel geprint worden.
- Hier en daar wordt in het programma gebruik gemaakt van een endless loop. Deze loop kan in C gedefinieerd worden door while(1) of door for(;;).
  #define LOOP while(1)

- De als CONSTANT gedeclareerde variabelen:

- MAX_AGE: Maximum leeftijd van de bladluizen (in dagen).
- RETENTION_TIME: Tijdsduur tussen het tijdstip waarop de bladluis het virus verwerft en het tijdstip waarop de bladluis haar infectiositeit verliest (in uren).
- ACQUISITION_FEEDING_TIME: Benodigde zuigtijd van een bladluis op een ziek blad van een bieteplant om het virus uit de plant op te nemen (in uren).

- TRANSMISSION_FEEDING_TIME: Benodigde zuigtijd van een infectieuze bladluis op een willekeurig blad van een bieteplant om het virus op de plant over te dragen (in uren).
- MIN_FLYING_TEMP: Laagste temperatuur waarbij de bladluizen gaan vliegen.
- MAX_FLYING_TEMP: Hoogste temperatuur waarbij de bladluizen vliegen.
- MIN_MOBILITY_TEMP: Respectievelijk de minimum, optimum en
- OPTIMUM_MOBILITY_TEMP: maximum temperatuur voor lopen.
- MAX_MOBILITY_TEMP:
- MIN_LEAF_APPEARANCE_TEMP: Laagste temperatuur waarbij nog nieuwe bladeren gevormd worden.
- MIN_LEAF_EXPANSION_TEMP: Laagste temperatuur waarbij nog bladexpansie plaatsvindt.
- OPTIMUM_GROWTH_TEMP: Respectievelijk de optimum en de maximum
- MAX_GROWTH_TEMP: temperatuur voor bieteplantengroei.
- OPTIMUM_MOBILITY_AGE: Leeftijd van de luis waarbij ze haar maximale loopsnelheid bereikt. (in uren).
- MAX_VELOCITY: Hoogste loopsnelheid, die alleen bereikt wordt door luizen met een leeftijd gelijk aan de OPTIMUM_MOBILITY_AGE en bij een temperatuur gelijk aan de OPTIMUM_MOBILITY_TEMP. (in cm/uur).

De temperatuur is altijd in graden Celcius.

- De rest van de globale variabelen:

- fieldlength: De lengte van het te simuleren veld in meters.
- fieldwidth: De breedte van het te simuleren veld in meters.
- lengthplants: Het aantal planten over een meter lengte.
- widthplants: Het aantal planten over een meter breedte.
- lengthfraction: Het aantal centimeters tussen de bieteplanten die over de lengte van veld staan.
- widthfraction: Het aantal centimeters tussen de bieteplanten die over de breedte van het veld staan.
- tot_lengthplants: Het totaal aantal planten in de lengte van het veld.
- tot_widthplants: Het totaal aantal planten in de breedte van het veld.
- tot_plants: Het totaal aantal planten in het bietenveld.
- tot_aphids: Het totaal aantal luizen op de bieteplanten.
- time: Bevat steeds het actuele simulatietijdstip (uren).
- maxtime: Het maximaal aantal te simuleren uren.
- month/day: De datum van aanvang van de simulatie.
- sowing_month/sowing_day: De zaaidatum van de bieten.
- monthdays[13]: Het aantal dagen van de maanden.
- monthname[13]: De namen van alle maanden van het jaar.
- field_counting[214]: Bevat veldtellingen van het aantal luizen op de "middelste" 39 planten van een bietenveld van 12 bij 12 meter. De periode waarover gesimuleerd kan worden (1_4_86 t/m 31_10_86) bevat 214 dagen. Voor elke dag is een veldtelling opgenomen. De gegevens worden ingelezen van één van de files luizen0.dat t/m luizen6.dat.

- incubation_time[31]: De incubatietijden van de bieteplanten, over de simulatietijdspanne van bijna 31 weken, per week. De gegevens voor laatgezaaide bieten worden gelezen van de file incyoung.dat. De gegevens voor vroeggezaaide bieten worden gelezen van de file incubold.dat. Vroeggezaaide bieten zijn gezaaid op 20 mei. Laatgezaaide bieten zijn gezaaid op 18 april.
- temperature: Bevat steeds de temperatuur behorende de bij het actuele tijdstip van de simulatie.
- temp[TEMP_GEG]: Temperatuurgegevens over de periode van 1_4_86 t/m 31_10_86. De temperatuurgegevens worden ingelezen van de file temp.dat. Per drie uur is er een waarde opgenomen.
- period: Is een getal dat wijst naar een plaats in de array van temperaturen. Period wordt tijdens de gehele simulatie om de drie uren opgehoogd zodat period altijd wijst naar de juiste plaats in de array.
  Voor elk drieuurssegment is namelijk een temperatuurwaarde in de array van temperaturen (temp[]) opgenomen.
- period_on_sowing_date: De waarde van period op het moment van zaaien van de bieten.
- growthless_period: Het beschrijvende model voor de groei van bieteplanten is geldig 20 dagen na het zaaien van de bieten. Deze periode wordt in het model de groeiloze periode genoemd.
- aphid_days: Het totale aantal luisdagen vanaf het begin van de simulatie.
- total_cdays: Het totaal aantal graaddagen dat verstreken is sinds de start van de simulatie.
- mean_temp: De gemiddelde temperatuur van de komende dag (24 uur). Van deze informatie wordt in het plantengroeimodel gebruik gemaakt.
- nbr_of_leaves: Het aantal bladeren dat zich aan de bieteplant bevindt.
- total_leaf_area: Het totale bladoppervlak (in cm*cm) per plant.
- growth_factor[MAX_LEAVES]: Factoren die voor elk blad afzonderlijk bepalen wat de uiteindelijke maximale afmetingen kunnen worden.
- final_leaf_area[MAX_LEAVES]: Bladoppervlak (in cm*cm) van elk afzonderlijk blad.
- stage_of_development[MAX_LEAVES]: Actueel ontwikkelingsstadium van elk afzonderlijk blad.
- simulation_data_from_file: Als boolean gebruikte variabele welke aangeeft of de waarden waarmee de simulatie opgestart wordt van file komen of interactief ingelezen moeten worden. De boolean zal 1 (true) zijn indien het programma met parms opgestart is. Anders 0 (false).
- aphid_data_from_file: Als boolean gebruikte variabele welke aangeeft of de gegevens over het aantal luizen in de simulatie van file komen of interactief ingelezen moeten worden.
- seed: Zaadje voor de uniforme verdeling.
- p: Filepointer. Wijst naar de file waar de informatie naartoe geschreven zal worden.
- dummy_aphid: Pointer naar een structuur van het type aphid. Deze pointer zal wijzen naar een vaste struct aphid welke gebruikt wordt om de double linked list in stand te houden. Zie ook delete_aphid().

- De diverse C functies (uit simulate.c):

Achter elke functie staan tussen haakjes de namen van de functies die de functie aanroepen. Van elke functie wordt omschreven waartoe ze dient. Als eerste worden de functies uit de file simulate.c beschreven.

- cast_skin() (main()):

Indien de luis (this_aphid) jonger is dan 145 uren, dan zijn nog niet alle vervellingen die een luis gedurende haar ontwikkeling meemaakt afgelopen. In dit model is vervellen temperatuur-onafhankelijk. De luizen vervellen om de anderhalve dag (36 uur) vier keer in totaal. Na zes dagen (144 uur) zijn alle vervellingen afgelopen.

- virus_exchange() (main())

In deze functie wordt gekeken of een besmette luis een niet besmette plant infecteert of het virus uit een besmette plant opneemt. Voor alle planten is het veld time_of_contamination geinitialiseerd op -1. In het begin van de functie wordt gekeken of het veld time_of_contamination van de plant, waar this_aphid zich op bevindt, groter of gelijk aan 0 is. Is dit zo dan is de plant besmet met het virus, is dit niet zo dan is de plant onbesmet. In beide gevallen moeten er een paar dingen gecontroleerd worden. Als de plant besmet is (field[x][y]->time_of_contamination >= 0), kan de luis (this_aphid) het virus uit de plant opnemen indien:

- de latentieperiode voorbij is
  (time - field[x][y]->time_of_contamination > latent_period())
- en de luis op een besmet blad zuigt (&& contaminated_leaf()).
- en de luis er lang genoeg zuigt om het virus op te nemen
  (&& time - this_aphid->time_of_switch > AQUISITION_FEEDING_TIME).

Is aan deze drie voorwaarden voldaan dan neemt de bladluis het virus op uit de bieteplant (this_aphid->time_of_contamination = time;). Wordt aan één van deze drie voorwaarden niet voldaan dan moet er uit de functie gesprongen worden (return;).
In het geval dat de plant niet met BYV besmet is, field[x][y]-> time_of_contamination is dan nog -1, kan de plant door de luis besmet worden indien:

- de luis drager is van het virus (aphid_contaminated()).
- en de luis lang genoeg zuigt op de plant om het virus over te
  dragen (&& time - this_aphid->time_of_switch >
  TRANSMISSION_FEEDING_TIME).

Is aan deze beide voorwaarden voldaan dan draagt de luis het virus over op de bieteplant (field[x][y]->time_of_contamination = time;). Het aantal bladeren wat zich op dit moment aan de plant bevindt, zal het virus niet krijgen behoudens het blad waaraan de plant besmet is. De oppervlakte van deze plek wordt buiten beschouwing gelaten in de simulatie. Dus: field[x][y]->healthy_leaves =

floor(nbr_of_leaves);. Alles wat de plant hierna nog aan blad erbij krijgt, zal besmet zijn met BYV. Een plant eenmaal besmet met BYV zal van een eventuele volgende besmetting door een bladluis geen effecten ondervinden. Een bladluis die reeds besmet was met BYV zal na het zuigen op een besmet blad het virus weer een volle lengte van RETENTION_PERIOD bij zich dragen.

- latent_period() (virus_exchange())

Deze functie berekent de lengte van de latentieperiode en levert deze waarde als functiewaarde op. Verondersteld wordt, dat de latentieperiode van 4 dagen in het jongste gewas (0 dagen) toeneemt tot een lengte van 12 dagen in het oudste gewas (7 maanden). De waarde moet opgeleverd worden in uren (return periode;). Periode wordt berekend uit:
(de leeftijd van het gewas in uren/maximale leeftijd van het gewas in uren) * 192.0 + 96.0.

- contaminated_leaf() (virus_exchange())

Deze functie moet een 1 opleveren als functiewaarde indien de luis zich op een besmet blad bevindt en een 0 indien niet. Om de klasse te berekenen waarin de zieke plant zich bevindt;

(besmette bladoppervlak * 10)/totale bladoppervlak

De uitkomst van deze berekening ligt tussen 0 en 10. Er zijn 11 besmettingsklassen, variërend van 0% ziek blad t/m 100% ziek blad (klasse 0 t/m 10). De kans dat een luis op een ziek blad zit is groter naarmate de klasse waarin de plant zich bevindt hoger is. Een plant die minder dan 10% besmet bladoppervlak heeft, komt in klasse 1 terecht.
Om het gebruik van kanstrekkingen zoveel mogelijk te vermijden, wordt hier van een andere methode gebruik gemaakt om te bepalen of de luis al dan niet op een besmet blad zuigt. De waarde van de klasse van de plant wordt opgeteld bij een restgetal. Komt de waarde van dit restgetal boven de 10 dan wordt besloten dat de luis op een besmet blad zit. Het restgetal wordt met een waarde van 10 verlaagd. De functie levert als waarde 1 op. Blijft na de optelling van de klasse bij het restgetal de waarde van het restgetal onder de 10, dan levert de functie als functiewaarde een 0 op. De waarde van het restgetal (fraction) blijft onveranderd. De waarde van het restgetal gaat niet verloren indien de functie verlaten wordt. Fraction is een statische variabele.

- aphid_contaminated() (virus_exchange())

Deze functie levert als functiewaarde een 1 op indien de luis infectieus is en een 0 indien niet. De luis verliest haar infectiositeit indien de tijdsduur verstreken tussen opname van het virus (this_aphid->time_of_contamination) en "nu" (time) groter is dan RETENTION_PERIOD (de retentie periode).

- symptoms_development() (print_image())

Functie om de mate van vergeling te bepalen. Het hele veld wordt hiertoe doorgerekend. De lengte van de incubatieperiode hangt af van het moment van infectie. De incubatieperioden zijn per week opgeborgen in de integerarray incubation_time[31]. De array wordt ingelezen van file. De maximale simulatieduur is bijna 31 weken. Met behulp van period, time en field[x][y]->time_of_contamination wordt berekend welke van de 31 elementen uit deze array gebruikt moet worden. De variabele index wordt hiervoor gebruikt.
Met behulp van period wordt bepaald welke waarde index zou moeten krijgen indien de plant dit uur besmet zou zijn met het virus. Period wijst naar een plaats in de tabel met temperatuurgegevens welke hoort bij het moment van simuleren. Omdat er 56 temperatuursgegevens per week opgenomen zijn, moet period gedeeld worden door 56 teneinde het weeknummer te verkrijgen. Indien de plant niet dit uur maar eerder in de simulatie besmet is met het virus, moet de lengte van de tussenliggende periode berekend worden in weken. Deze wordt berekend met behulp van de tijd van besmetting van de bieteplant (field[x][y]->time_of_contamination) en de huidige simulatietijd (time). Dit aantal weken moet afgetrokken worden van het huidige weeknummer om het weeknummer te krijgen van de week waarin de plant geinfecteerd is:

index =
period/56.0 - (time-field[x][y]->time_of_contamination)/168.0

Hierna wordt bekeken of de incubatieperiode verstreken is: (time - field[x][y]->time_of_contamination > incubation_time). Zo ja, dan is de plant vergeeld (field[x][y]->yellowed = 1;). Deze berekeningen worden alleen uitgevoerd indien de plant niet reeds vergeeld is en al wel met BYV besmet (field[x][y]->yellowed == 0 && field[x][y]-> time_of_contamination >= 0)!

- distance_covered() (main())

Functie voor het berekenen van de afgelegde afstand voor elke bladluis in een uur tijd. De afgelegde afstand is afhankelijk van de loopfrequentie en de loopsnelheid. De afgelegde afstand in de x en de y richting is afhankelijk van de hoek waaronder de luis loopt. De loopfrequentie wordt allereerst bepaald. Indien het aantal bladeren aan de planten kleiner is dan 8 wordt het percentage berekend met de volgende vergelijking:

percentage = 7.75 * nbr_of_leaves
Indien het aantal bladeren groter is dan 8 met deze vergelijking:
percentage = -0.41 * nbr_of_leaves

Indien het gewas nog niet gesloten is, het aantal bladeren is dan nog kleiner dan 12, loopt slechts 1% van de bladluizen. Voor het lopen van de bladluizen wordt nu de loopsnelheid bepaald (velocity) en en een correctiefactor voor temperatuursinvloeden (factor).
Uit de bovenberekende waarden kan nu de afgelegde afstand voor dit uur berekend worden. Daar de loopsnelheid berekend wordt in cm/uur hoeven we niet met het aantal uren te vermenigvuldigen (de afstand wordt immers elk uur berekend). total_distance = velocity * factor * percentage;

Vervolgens wordt de hoek waaronder de luis loopt bepaald en aan de hand daarvan de afgelegde afstand in de x_ en de y_richting. Aangezien verondersteld wordt dat lopende luizen het veld niet kunnen verlaten, moeten luizen die het veld uit willen lopen aan de grenzen van het veld tot staan gebracht worden. Lopen de luizen niet te ver dan dienen de coördinaten van de luis aangepast te worden aan zijn nieuwe positie in het veld.
Indien een luis gelopen heeft, moet het veld time_of_switch gelijk gemaakt worden aan time, de actuele simulatietijd.

- aphid_removal() (main())

In deze functie worden een bladluis verwijderd:
- Indien een luis gevleugeld is en de temperatuur en de leeftijd van de luis laten het vliegen toe;
- Indien een luis zijn maximum leeftijd bereikt heeft;
- Indien een luis gepredeert wordt.

In het model wordt het aantal bladluizen gereguleerd aan de hand van in het veld uitgevoerde bladluistellingen. Het verwijderen van luizen mag geen invloed hebben op het aantal bladluizen in de simulatie. In deze functie verwijderde bladluizen worden daarom vervangen door nieuwe met een leeftijd van 0 uur. Deze functie beïnvloedt de leeftijdsverdeling van de bladluizen in de simulatie. Eén procent van de bladluizen wordt op uurbasis gepredeerd. Op een dag wordt dus 24% van de luizen gepredeerd.

- plant_growth() (main())

Deze functie beschrijft de groei van bietebladeren in de tijd als een functie van de temperatuur. De bladverschijning en de bladgroei per blad worden berekend alsmede het totale bladoppervlak, het totale aantal bladeren aan de plant en het oppervlak per blad.

- Bladverschijningssnelheid in bladeren per dag:
  3.16e-2 * temperatuur - 3.16e-2.

- Bladgroei in cm^2/dag:
  groeifactor * (1.1 * temperatuur - 3.3).

- Ontwikkelingsduur in graaddagen:
  groeifactor * 4.25 * temperatuur + 355.

Niet alle bladeren worden even groot. Het grootst worden de bladeren rond het 10^e blad. In dit model is de uiteindelijke afmeting van het blad afhankelijk van bladgroeisnelheid en ontwikkelingsduur.

- weather() (main())

Door deze functie wordt om de 24 uur de gemiddelde temperatuur (van 0.00 uur tot 24.00 uur) en het aantal luisdagen bepaald. Om de drie uur wordt de globale variabele temperature gevuld met de juiste waarde en het totale aantal graaddagen wat verstreken is sedert de start van de simulatie wordt bijgewerkt. Period wordt met één opgehoogd.

- print_image() (main())

In deze functie wordt de output naar scherm en naar de verschillende outputfiles verzorgt. Voorlopig geen uitleg bij deze functie.

- aphid_number_checking() (main())

In het verslag van Corné Kempenaar staat beschreven hoe de veldtellingen zijn gedaan en hoe de veldtellingen zijn omgewerkt naar een schatting van het aantal bladluizen op de middelste 39 planten in het veld. In deze functie wordt bekeken of het aantal luizen dat zich op de middelste 39 planten in de simulatie bevindt, overeenkomt met het aantal bladluizen op de zelfde 39 planten in het veld. De veldtellingen zitten in de array field_counting[214] en worden van file ingelezen. De maximale simulatieduur is 214 dagen. Voor elke dag is in deze array een waarde opgenomen. Indien het verschil tussen het aantal in de simulatie getelde luizen en het aantal in het veld getelde luizen niet groter is dan 10%, hoeft het aantal luizen in de simulatie niet aangepast te worden. Indien het verschil wel groter is dan 10%, dan moet het aantal luizen zo veranderd worden dat het aantal luizen op de middelste 39 planten in de simulatie gelijk is aan het aantal dat in het veld op de overeenkomstige 39 planten geteld is. Deze luizen-aantalbijstelling vindt plaats in de functie aphid_number_adjusting().

- aphid_number_adjusting() (aphid_number_checking())

In deze functie wordt het aantal bladluizen gelijkelijk over het gehele veld verhoogd of verlaagd zo, dat het aantal bladluizen op de middelste 39 planten in de simulatie overeenkomt met het aantal dat voor deze planten in het veld geschat is. Daartoe moet een factor berekend worden welke het verschil aangeeft tussen bovenbedoelde aantallen. Factor wordt zo berekend dat de volgende vergelijking geldt:

Het aantal in het veld getelde bladluizen = het aantal in de simulatie getelde bladluizen * factor.

Is het aantal getelde bladluizen in de simulatie gelijk aan 0 dan wordt aangenomen dat er luizen met een leeftijd van 240 uren uitgezet moeten worden op de middelste plant in het veld van de simulatie (net zoals in de veldproeven van Corné Kempenaar). Is het aantal getelde bladluizen in het veld 0, dan moeten alle luizen uit de simulatie verwijderd worden. De waarde van factor kan nooit gelijk zijn aan 1. Evenmin kunnen het aantal bladluizen in de simulatie en het aantal bladluizen in het veld beide gelijk aan 0

zijn. Zijn het aantal bladluizen in de simulatie en het aantal bladluizen in het veld beide ongelijk aan 0 dan moet er altijd een correctieslag uitgevoerd worden.
Indien het aantal bladluizen op een plant niet gelijk is aan nul (field[x][y]-> nbr_of_aphids != 0), wordt met behulp van factor berekend hoeveel bladluizen er op de plant moeten komen (aphids_calculated). Aphids_calculated bevat een real getal. Het deel achter de komma wordt bewaard in fraction en later opgeteld bij aphids_calculated (bezemen). Indien het overgebleven berekende integer aantal bladluizen verschilt van het aantal bladluizen op de plant, moet het aantal bladluizen aangepast worden.
Zolang het aantal berekende bladluizen groter is dan het aantal bladluizen op de plant worden er nieuwe luizen gecreëerd en op de plant bijgezet. Zolang het aantal berekende bladluizen kleiner is, moeten er bladluizen verwijderd worden van de betreffende plant. Hiertoe moet er in de double linked list van luizen gezocht worden naar luizen waarvan de coördinaten overeenkomen met die van de plant.

- date() (print_image() en aphid_number_checking())
In deze functie wordt aan de hand van de datum waarop de simulatie gestart is en het aantal uren dat sinds de start verstreken is, berekend wat de datum van dit moment is. De datum wordt naar file geschreven. De variabele tijd wordt geïnitialiseerd met de waarde van time, dag met day en maand met month. Zolang het aantal uren in de variabele tijd groter is dan het aantal uren dat de maand vanaf dag bevat, wordt het aantal maanden met 1 opgehoogd en het aantal uren uit deze maand van tijd afgetrokken. Dag wordt 1. Dit wordt herhaald totdat het aantal uren in tijd hiervoor te klein is geworden. Daarna wordt de rest van het aantal uren in tijd gedeeld door 24 en opgeteld bij het aantal dagen (dag). Het aantal uren wat dan nog rest geeft de tijd van de dag aan (tijd).

- rerun() (main())

In rerun() wordt geheugenruimte vrijgemaakt voor een eventuele volgende simulatie. Rerun() wordt aangeroepen na elke simulatie indien de simulatiegegevens van file gelezen worden. Alleen dan immers kunnen er meerdere simulaties achtereen gedaan worden. Alle structuren aangemaakt voor de bieteplanten in het veld en alle structuren van de bladluizen worden "weggegooid". Time en total_cdays evenals aphid_days en alle variabelen die gegevens over de bieteplanten bevatten, worden teruggezet op 0.

- main() (het hoofdprogramma!)

Main() wordt aangeroepen met twee argumenten, argc en argv. Argc is een integer waarin staat hoeveel parms er bij de aanroep van het programma meegegeven zijn. De programmanaam zelf telt ook mee! Argv is een pointer naar een pointer naar een reeks characters oftewel een array van pointers naar reeksen van characters. Elke pointer uit de array wijst naar een characterstring waarin zich de naam van een parm bevindt. Er zijn zoveel pointers als er parms achter de programmanaam staan plus een pointer naar de programmanaam zelf (argv[0]).

Indien argc groter is dan 1 betekent dit dus dat er achter de programmanaam parms (zie bijlage: Handleiding voor het gebruik van het simulatieprogramma BYV) zijn meegegeven. In dat geval moeten alle simulatiegegevens van file ingelezen worden. De globale, als short gedeclareerde, variabele simulation_data_from_file wordt daartoe op 1 (true) gezet. Er worden zoveel simulaties gemaakt als er parms achter de programmanaam staan.

Voor elke simulatie:
-1-: wordt een outputfile gemaakt waarin de gegevens van de simulatie weggeschreven worden. De outputfiles worden genaamd sim1.dat, sim2.dat enz.
-2-: wordt init_dialog() aangeroepen met als argument een pointer naar de naam van de file waaruit de opstartgegevens van de simulatie gelezen moeten worden. In init_dialog() zal deze file geopend en gelezen worden. Na het doorlopen van de functie init_dialog() zijn alle opstartwaarden voor de simulatie ingelezen en kan de simulatie beginnen.
-3-: wordt de variabele time vanaf 0 telkens met een opgehoogd totdat de maximale tijdsduur (maxtime) bereikt is. Voor elke waarde van time wordt het volgende gedaan:
- aanroepen van de functie weather();
- indien print_interval_time een geheel aantal malen uit time gehaald kan worden, wordt print_image() aangeroepen.
- om de 24 uur wordt plant_growth() aangeroepen.
- indien de luisgegevens van file komen, wordt eveneens om de 24 uur aphid_number_checking() aangeroepen.
- de double linked list van luizen wordt doorlopen waarbij this_aphid telkens naar een volgende luis wijst. Voor elke luis worden de volgende functies aangeroepen:
--cast_skin();
--virus_exchange();
--distance_covered();
--aphid_removal();

-4-: wordt de outputfile, nadat time gelijk geworden is aan maxtime, gesloten. Indien de simulatiegegevens van file komen en i kleiner is dan argc, wordt rerun() aangeroepen zodat daarna een nieuwe simulatie opgestart kan worden.

Als alle simulaties afgelopen zijn, wordt het scherm schoon gemaakt en het programma beëindigd.

- Enkele C functies uit de file readdata.c:

- openen() (init_dialog(), init_var() en main())

De functie openen() is geschreven om een waarschuwing te geven als een file geopend moet worden maar de file niet geopend kan worden. Er wordt gewacht totdat er een willekeurige toets aangeslagen wordt. In de tussentijd heeft de gebruiker de gelegenheid om de goede disk in de juiste diskdrive te stoppen. Na het aanslaan van de willekeurige toets wordt opnieuw geprobeerd om de betreffende file te openen.

- uniform() (distance_covered(), aphid_number_adjusting() en put_aphids())

Deze functie levert bij aanroep als functiewaarde een integer random getal op met een waarde vanaf 0 tot max, waar max de maximum waarde is welke door de gebruiker bij de aanroep van uniform() in het interface meegegeven wordt. Let op! De waarde max wordt zelf nooit als functiewaarde afgelevert. De waarde 0 wel. De verdeling van de random getallen is uniform.

- scan_input() (init_dialog(), contaminate_plant(), put_aphids() en change_constants())

In scan_input() worden getallen, letters en alle andere door de gebruiker ingetoetste tekens gecontroleerd op de geldigheid van hun voorkomen op de plaats van inlezen. Wordt er bijvoorbeeld een positief integer getal verwacht op de plaats van inlezen, dan controleert deze functie of de ingetoetste tekens wel alle cijfers zijn. Wordt er een foute toets aangeslagen dan wordt hierop een foutmelding gegeven. Het bewuste teken verschijnt dan niet op het scherm. Voor het inlezen van een integer getal moet deze functie aangeroepen worden met een "I", voor het inlezen van een float getal met een "F" en voor het inlezen van een enkele letter met een "C". Het maximale aantal letters dat per plaats ingelezen kan worden is altijd één. Voor integer en float getallen geldt ook dat er slechts een beperkt aantal tekens per inleesplaats ingetoetst mogen worden met een maximum van 8. Bij het aanroepen van deze functie moet in het interface meegegeven worden hoeveel tekens er maximaal ingelezen mogen worden op de betreffende inleesplaats. Wordt het aantal ingetoetste tekens wat toegestaan is overschreden of bereikt, dan wordt de functie beëindigd. Bij het inlezen van een carriage return wordt de functie eveneens beëindigd. De functie levert na beëindiging als functiewaarde een pointer op naar de string waarin de ingetoetste tekens staan. De string bevat dezelfde tekens als zichtbaar geworden zijn op het scherm. De plaats op het scherm waar een waarde ingelezen moet worden, wordt in het interface meegegeven met row en col (column) welke de rij en de kolom op het beeldscherm aangeven. Indien bij de aanroep van scan_value() op de plaats van error een 1 wordt meegegeven in het interface zal het gebied van het scherm waar de foutmeldingen verschijnen leeg gemaakt worden. Indien er bij het inlezen van een waarde een punt (.) gebruikt is, is *point 1 (true) gemaakt. De waarde van point is later belangrijk voor het berekenen van de waarde van de string (waar deze functie de pointer naar oplevert).

- value() (init_dialog(), contaminate_plant(), put_aphids() en change_constants())

In value wordt de waarde van de string, waar scan_input() de pointer naar oplevert, bepaald. Deze waarde wordt als (float) functiewaarde opgelevert. Deze functie is geschreven omdat de standaardfuncties atoi() en atof() niet voldeden.

- create_aphid() (put_aphids(), aphid_number_adjusting() en aphid_removal())

Deze functie ruimt geheugenruimte in voor een structuur van het type aphid. De velden van deze structuur krijgen de waarden welke als parameters met de functieaanroep meegegeven zijn. De nieuw gecreëerde structuur wordt opgenomen vooraan de double linked list van luizen. De luis wordt bijgeteld bij het totale aantal luizen en bij het aantal luizen van de plant waar ze, volgens de in het interface meegegeven coördinaten, op terecht komt.

- delete_aphid() (aphid_number_adjusting() en aphid_removal())

In delete_aphid() wordt de struct aphid waar de pointer this_aphid naar wijst, uit de double linked list van luizen verwijderd. De geheugenruimte die deze structuur inneemt, wordt weer vrijgemaakt. Het totale aantal luizen wordt verminderd met één en het aantal luizen van de plant waar ze zich op bevond evenzo. De double linked list van luizen wordt hersteld en this_aphid wijst hierna naar de luis die voor de verwijderde luis in de double linked list stond. Betrof het de eerste luis uit de double linked list die verwijderd moest worden, dan zal er een nieuwe first_aphid zijn. Dummy_aphid->next zal naar de nieuwe first_aphid wijzen en this_aphid naar dummy_aphid. De nieuwe first_aphid->pred zal naar NULL wijzen. Betrof het de laatste luis uit de double linked list die verwijdert moest worden, dan zal de luis voor de laatste luis de nieuwe laatste luis worden. De nieuwe last_aphid->next zal naar NULL wijzen en this_aphid naar de nieuwe last_aphid. Stond er slechts één luis in de double linked list dan worden first_aphid en last_aphid beide NULL. Dummy_aphid->next zal ook naar NULL wijzen en this_aphid naar dummy_aphid.

- init_var() (init_dialog())

In init_var() worden wat globale variabelen geïnitialiseerd welke niet in init_dialog(), contaminate_plant(), put_aphids() of change_constants() geïnitialiseerd worden.

- De lengte van de groeiloze periode van de planten (de periode tussen zaaien en het geldig worden van het plantgroeimodel) wordt geïnitialiseerd op 20 dagen.
- De dummy_aphid krijgt een structuur om naar te wijzen.
- De temperaturentabel wordt ingelezen van de file temp.dat.
- De incubatietijden worden ingelezen van de file incyoung.dat of incubold.dat.
- De groeifactoren van alle bladeren worden geïnitialiseerd.
- Het aantal dagen dat ligt tussen de begindatum van de tabel met de temperatuurwaarden (1 april '86) en het moment van zaaien van de bieten, wordt berekend. Aan de hand van dit aantal dagen worden period en period_on_sowing_date geïnitialiseerd.
- Hierna wordt het aantal dagen dat ligt tussen het moment van zaaien en de startdatum van de simulatie berekend. Indien het aantal dagen groter is dan de groeiloze periode moeten de planten groeien voordat de simulatie kan beginnen. Period, het aantal dagen dat de planten zullen moeten groeien (dagen) en growthless_period moeten bijgesteld worden. Is het aantal

berekende dagen tussen het moment van zaaien en de startdatum van de simulatie kleiner dan de groeiloze periode dan hoeven de planten dus niet te groeien. Period en growthless_period worden bijgesteld en dagen (het aantal dagen dat de bieteplanten moeten groeien voordat de simulatie mag beginnen) wordt nul.

- Tot slot moeten de planten groeien gedurende het aantal dagen dat hierboven uitgerekend is (dagen). Daarvoor moeten de functies weather() en plant_growth() aangeroepen worden.

Bijlage F. Temperaturentabel

-1	2	5	8	8	5	2	-1	1 april
-1	2	6	9	9	6	2	-1	
-2	-1	5	9	9	5	-1	-2	
0	2	6	8	8	6	2	0	
-1	1	5	8	8	5	1	-1	
0	3	6	9	9	6	3	0	
1	3	6	7	7	6	3	1	
2	3	5	6	6	5	3	2	
1	3	5	7	7	5	3	1	
-1	1	5	8	8	5	1	-1	10 april
-9	-4	3	8	8	3	-4	-9	
-9	-4	4	9	9	4	-4	-9	
-6	-1	6	10	10	6	-1	-6	
-1	2	5	8	8	5	2	-1	
2	5	9	12	12	9	5	2	
4	7	11	15	15	11	7	4	
1	5	11	15	15	11	5	1	
-2	2	8	12	12	8	2	-2	
-2	2	8	11	11	8	2	-2	
-1	3	8	11	11	8	3	-1	20 april
2	5	10	14	14	10	5	2	
1	5	11	14	14	11	5	1	
0	5	12	16	16	12	5	0	
0	5	12	17	17	12	5	0	
3	7	13	17	17	13	7	3	
4	8	13	16	16	13	8	4	
8	10	13	15	15	13	10	8	
4	7	10	13	13	10	7	4	
2	6	11	15	15	11	6	2	
1	6	13	18	18	13	6	1	30 april
0	8	16	21	21	16	8	0	1 mei
8	14	20	24	24	20	14	8	
8	13	21	26	26	21	13	8	
8	10	16	21	21	16	10	8	
3	9	17	22	22	17	9	3	
3	8	14	17	17	14	8	3	
5	9	15	19	19	15	9	5	
4	8	13	16	16	13	8	4	
7	10	14	17	17	14	10	7	
9	11	15	18	18	15	11	9	10 mei
6	9	13	15	15	13	9	6	
7	11	17	22	22	17	11	7	
3	8	15	18	18	15	8	3	
6	9	14	19	19	14	9	6	
6	8	11	13	13	11	8	6	
2	7	13	17	17	13	7	2	
1	8	17	21	21	17	8	1	
5	10	17	20	20	17	10	5	
7	12	19	24	24	19	12	7	
8	14	21	25	25	21	14	8	20 mei
9	13	18	21	21	18	13	9	
9	13	16	18	18	16	13	9	
8	10	14	16	16	14	10	8	
5	9	14	17	17	14	9	5	

```
 5 10 17 21 21 17 10  5
 5 11 18 22 22 18 11  5
 5 10 15 19 19 15 10  5
 4  9 15 18 18 15  9  4
 3  7 13 17 17 13  7  4
-1  6 13 17 17 13  6 -1        30 mei
-1  6 15 20 20 15  6 -1
 4  9 12 14 14 12  9  4         1 juni
 3  9 17 22 22 17  9  3
 2  8 13 15 15 13  8  2
 5  8 11 13 13 11  8  5
 6  8  9 10 10  9  8  6
 6  9 11 12 12 11  9  6
 6  8 10 11 11 10  8  6
 6  9 13 16 16 13  9  6
 6 12 18 22 22 18 12  6
 8 13 20 24 24 20 13  8        10 juni
 7 10 14 17 17 14 10  7
 6  9 15 19 19 15  9  6
 1  8 16 20 20 16  8  1
 5 12 19 22 22 19 12  5
11 16 22 25 25 22 16 11
14 19 25 28 28 25 19 14
14 19 24 27 27 24 19 14
 4 10 17 20 20 17 10  4
 3 10 17 21 21 17 10  3
10 13 18 22 22 18 13 10        20 juni
 9 12 16 19 19 16 12  9
10 15 20 24 24 20 15 10
 8 15 22 26 26 22 15  8
 6 11 16 17 17 16 11  6
 5 12 20 24 24 20 12  5
11 16 21 25 25 21 16 11
12 17 23 27 27 23 17 12
 8 14 22 26 26 22 14  8
10 15 22 26 26 22 15 10
12 17 24 27 27 24 17 12        30 juni
14 19 25 28 28 25 19 14         1 juli
14 18 23 26 26 23 18 14
13 17 21 24 24 21 17 13
12 15 19 22 22 19 15 12
13 15 18 20 20 18 15 13
 8 13 18 21 21 18 13  8
 7 11 17 21 21 17 11  7
 7 12 17 21 21 17 12  7
10 12 16 19 19 16 12 10
 9 12 16 20 20 16 12  9        10 juli
 7 11 16 20 20 16 11  7
 6 10 17 22 22 17 10  6
 6 11 16 19 19 16 11  6
11 14 19 23 23 19 14 11
 9 14 21 26 26 21 14  9
10 16 24 29 29 24 16 10
 7 13 20 23 23 20 13  7
 6 11 17 21 21 17 11  6
 4 10 17 22 22 17 10  4
 4 10 17 21 21 17 10  4        20 juli
10 14 19 24 24 19 14 10
```

```
 9 12 17 20 20 17 12  9
 6 11 17 22 22 17 11  6
 8 11 15 18 18 15 11  8
10 13 15 16 16 15 13 10
10 14 18 20 20 18 14 10
11 15 20 22 22 20 15 11
13 17 23 26 26 23 17 13
13 16 20 22 22 20 16 13
 9 14 19 23 23 19 14  9        30 juli
 8 14 19 21 21 19 14  8
10 13 18 21 21 18 13 10         1 augustus
 4 12 20 23 23 20 12  4
 4 13 22 27 27 22 13  4
12 15 19 21 21 19 15 12
 5 10 17 19 19 17 10  5
 4 11 19 23 23 19 11  4
10 14 19 22 22 19 14 10
 8 12 16 18 18 16 12  8
 5 11 17 20 20 17 11  5
 8 12 19 23 23 19 12  8        10 augustus
 9 13 20 24 24 20 13  9
 7 13 19 22 22 19 13  7
 6 11 18 23 23 18 11  6
 7 14 22 26 26 22 14  7
10 14 19 22 22 19 14 10
10 13 18 21 21 18 13 10
 6 11 17 20 20 17 11  6
 7 11 18 22 22 18 11  7
11 13 16 19 19 16 13 11
 4  9 15 17 17 15  9  4        20 augustus
 5 10 16 20 20 16 10  5
11 13 17 19 19 17 13 11
 6 10 15 18 18 15 10  6
 5 10 15 18 18 15 10  5
 5 10 15 19 19 15 10  5
10 13 16 19 19 16 13 10
 9 10 11 12 12 11 10  9
 8 10 13 16 16 13 10  8
 8 10 14 16 16 14 10  8
10 12 15 17 17 15 12 10        30 augustus
 4  9 13 16 16 13  9  4
 6 10 14 17 17 14 10  6         1 september
 9 12 15 17 17 15 12  9
10 12 14 16 16 14 12 10
 5  9 14 17 17 14  9  5
 6  9 14 17 17 14  9  6
 6 10 13 15 15 13 10  6
 1  7 13 16 16 13  7  1
 4  8 12 14 14 12  8  4
 2  6 13 17 17 13  6  2
-1  4 11 16 16 11  4 -1        10 september
-1  4 10 15 15 10  4 -1
 0  6 12 17 17 12  6  0
-1  6 13 16 16 13  6 -1
 6 10 14 16 16 14 10  6
 2  6 10 11 11 10  6  2
 3  7 12 15 15 12  7  3
 4  8 12 15 15 12  8  4
```

1	6	12	16	16	12	6	1
0	5	12	17	17	12	5	0
0	6	13	17	17	13	6	0
4	9	14	18	18	14	9	4
9	11	14	15	15	14	11	9
5	10	15	18	18	15	10	5
4	8	14	18	18	14	8	4
1	6	12	16	16	12	6	1
0	5	12	17	17	12	5	0
3	7	13	17	17	13	7	3
4	8	14	18	18	14	8	4
8	11	16	20	20	16	11	8
5	10	14	15	15	14	10	5
2	8	17	23	23	17	8	2
4	9	15	18	18	15	9	4
2	7	14	20	20	14	7	2
1	7	14	18	18	14	7	1
8	12	18	23	23	18	12	8
7	11	17	20	20	17	11	7
7	12	17	21	21	17	12	7
3	8	15	20	20	15	8	3
4	9	16	20	20	16	9	4
8	12	17	21	21	17	12	8
5	10	14	16	16	14	10	5
11	13	17	20	20	17	13	11
6	11	16	20	20	16	11	6
11	14	19	23	23	19	14	11
10	14	19	23	23	19	14	10
7	10	14	15	15	14	10	7
5	8	14	18	18	14	8	5
9	11	15	18	18	15	11	9
5	8	11	13	13	11	8	5
6	8	12	15	15	12	8	6
6	8	10	12	12	10	8	6
6	9	12	14	14	12	9	6
3	5	8	11	11	8	5	3
1	5	9	12	12	9	5	1
7	9	11	13	13	11	9	7
3	6	9	11	11	9	6	3
3	6	11	14	14	11	6	3
10	11	12	13	13	12	11	10
3	7	11	14	14	11	7	3
4	7	10	13	13	10	7	4
5	7	11	13	13	11	7	5

20 september

30 september

1 oktober

10 oktober

20 oktober

30 oktober

De datum is in de originele file niet aanwezig.

Bijlage G. Bladluisaantallen

luizen1.dat

```
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0 464 414
364 314 264 214 164 122 114 105  97  88
 80  70  77  83  90  96 104 110 101  93
 86  77  69  60  50  43  36  29  22  15
  9   8   7   6   5   4   3   2   1   1
  1   1   1   1   1   2   2   2   2   2
  2   2   2   1   1   1   1   1   1   1
  1   1   1   1   1   1   1   1   1   1
  1   1   1   1   1   1   1   1   1   1
  1   1   1   1   1   1   1   1   1   1
  1   1   1   1   1   1   1   1   1   1
  1   1   1   1   1   1   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0
```

luizen2.dat

```
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0  77  68
 59  51  42  33  24  15  15  14  14  14
 13  13  18  23  27  32  37  42  37  32
 28  23  18  13   9   8   7   5   4   3
  2   2   2   2   2   1   1   1   1   1
  1   1   1   1   1   2   2   2   2   2
  2   2   2   1   1   1   1   1   1   1
  1   1   1   1   1   1   1   1   1   1
  1   1   1   1   1   1   1   1   1   1
  1   1   1   1   1   1   1   1   1   1
  1   1   1   1   1   1   1   1   1   1
  1   1   1   1   1   1   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0
```

luizen3.dat

```
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0  12  12
 11  11  11  10  10  10   9   8   7   6
  5   4   6   8  10  12  14  16  16  15
 15  14  14  13  13  11   9   7   5   3
  1   1   1   1   1   1   1   1   1   1
  1   1   1   1   1   2   2   2   2   2
  2   2   2   1   1   1   1   1   1   1
  1   1   1   1   1   1   1   1   1   1
  1   1   1   1   1   1   1   1   1   1
  1   1   1   1   1   1   1   1   1   1
  1   1   1   1   1   1   1   1   1   1
  1   1   1   1   1   1   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0
```

luizen4.dat

```
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0 488 446
404 362 320 278 236 194 183 172 160 148
137 126 137 148 159 170 181 192 184 176
168 160 152 144 136 117  99  80  62  43
 24  21  19  17  15  13  11   9   7   5
  5   6   7   8   8   9  10  11  11  12
 12  11  11  10  10  10   9   9   8   8
  8   7   7   6   6   6   5   5   4   4
  4   3   3   2   2   2   1   1   1   1
  1   1   1   1   1   1   1   1   1   1
  1   1   1   1   1   1   1   1   1   1
  1   1   1   1   1   1   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0
```

luizen5.dat

```
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0  94  86
 79  71  64  56  48  41  57  73  89 106
122 138 164 189 215 240 266 291 268 245
222 199 176 153 129 112  95  78  61  44
 27  24  22  20  17  15  12  10   7   5
  5   6   7   8   8   9  10  11  11  12
 12  11  11  10  10  10   9   9   8   8
  8   7   7   6   6   6   5   5   4   4
  4   3   3   2   2   2   1   1   1   1
  1   1   1   1   1   1   1   1   1   1
  1   1   1   1   1   1   1   1   1   1
  1   1   1   1   1   1   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0
```

luizen6.dat

```
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0  12  12
 12  12  11  11  11  11  15  20  24  28
 33  37  46  55  65  74  83  92  91  90
 89  89  88  87  86  72  58  44  30  16
  2   2   2   3   3   3   4   4   4   5
  5   6   7   8   8   9  10  11  11  12
 12  11  11  10  10  10   9   9   8   8
  8   7   7   6   6   6   5   5   4   4
  4   3   3   2   2   2   1   1   1   1
  1   1   1   1   1   1   1   1   1   1
  1   1   1   1   1   1   1   1   1   1
  1   1   1   1   1   1   0   0   0   0
  0   0   0   0   0   0   0   0   0   0
  0   0   0   0
```

Bijlage H. Incubatietijden

incubold.dat

```
1 april
  20   20   20   20   20
6 mei
  22   23   24   22
3 juni
  18   18   21   28
1 juli
  33   36   38   43   49
5 augustus
  62   69   70   70
2 september
  70   70   70   70   70
7 oktober
  70   70   70   70
```

incyoung.dat

```
1 april
  17   17   17   17   17
6 mei
  17   17   17   17
3 juni
  17   17   17   20
1 juli
  22   23   26   35   42
5 augustus
  51   55   60   60
2 september
  60   60   60   60   60
7 oktober
  60   60   60   60
```

De data staan er in de originele datafiles niet bij.

C SOURCE CODE

File readdata.c

```
#include <stdio.h>
#include <ctype.h>
#include <math.h>
#include <malloc.h>
#include <string.h>
#include <stdlib.h>
#include <color.h>

#define POWER(x) (x)*(x)
#define CONSTANT int
#define MAXLENGTH 100
#define MAXWIDTH 50
#define TEMP_GEG 1713
#define MAX_LEAVES 80
#define MAX_APHIDS 1000
#define MAX_CHAR 8
#define EMPTY_STRING "    (70 spaties)    "
#define LOOP while(1)


extern    CONSTANT MAX_AGE;
extern    CONSTANT RETENTION_TIME;
extern    CONSTANT ACQUISITION_FEEDING_TIME;
extern    CONSTANT TRANSMISSION_FEEDING_TIME;
extern    CONSTANT MIN_FLYING_TEMP;
extern    CONSTANT MAX_FLYING_TEMP;
extern    CONSTANT MIN_MOBILITY_TEMP;
extern    CONSTANT OPTIMUM_MOBILITY_TEMP;
extern    CONSTANT MAX_MOBILITY_TEMP;
extern    CONSTANT MIN_LEAF_APPEARANCE_TEMP;
extern    CONSTANT MIN_LEAF_EXPANSION_TEMP;
extern    CONSTANT OPTIMUM_GROWTH_TEMP;
extern    CONSTANT MAX_GROWTH_TEMP;
extern    CONSTANT OPTIMUM_MOBILITY_AGE;
extern    CONSTANT MAX_VELOCITY;


extern struct aphid {
  int time_of_birth;
  int time_of_contamination;
  int time_of_switch;
  int x, y;
  short winged;
  struct aphid *next, *pred;
} *first_aphid, *last_aphid;

extern struct beet {
  int class;
  int time_of_contamination;
  int nbr_of_aphids;
  int healthy_leaves;
  short yellowed;
} *field[MAXLENGTH][MAXWIDTH];
```

```
extern   int fieldlength, fieldwidth;
extern   float widthplants, lengthplants;
extern   int widthfraction, lengthfraction;
extern   int tot_widthplants, tot_lengthplants;
extern   int tot_plants, tot_aphids;
extern   int time, maxtime;
extern   int month, day;
extern   int sowing_month, sowing_day;
extern   int monthdays[13];
extern   int temperature, temp[TEMP_GEG];
extern   int print_interval_time;
extern   int field_counting[215];
extern   int incubation_time[31];
extern   int period_on_sowing_date;
extern   int period_on_starting_date;
extern   int growthless_period;
extern   int initial_age;
extern   int aphid_days;
extern   int predation;
extern   int interval;
extern   int period;

extern   float total_cdays;
extern   float mean_temp;
extern   float nbr_of_leaves;
extern   float total_leaf_area;
extern   float growth_factor[MAX_LEAVES];
extern   float final_leaf_area[MAX_LEAVES];
extern   float stage_of_development[MAX_LEAVES];

extern   short parmfile;
extern   short print_data;
extern   short print_picture;
extern   short print_luizengrafiek;
extern   short print_plantengrafiek;
extern   short simulation_data_from_file;
extern   short aphid_data_from_file;
extern   short there_were_no_aphids_yet;
extern   char  *monthname[13];
extern   char  parmfilename[16];
extern   int   seed;

extern   struct aphid *this_aphid;
extern   struct aphid *dummy_aphid;

extern   FILE *fopen();
extern   FILE *p1, *p2, *p3, *p4, *par;


extern   FILE     *openen(char *,char *);
extern   int      uniform(int);
extern   char     *scan_input(char *,short,short,short,short,short *);
extern   float    value(char *,short);
extern   void     screen(void);
extern   void     screen0(void);
extern   void     screen1(void);
extern   void     screen2(void);
```

```
extern   void     screen3(void);
extern   void     screen4(void);
extern   void     screen5(void);
extern   void     screen6(void);
extern   void     screen7(void);
extern   void     screen8(void);
extern   void     create_aphid(int,short,int,int,short);
extern   void     delete_aphid(void);
extern   void     init_dialog(char *);
extern   void     contaminate_plant(FILE *);
extern   void     put_aphids(FILE *);
extern   void     change_constants(FILE *);
extern   void     novalue(void);
extern   short    terminate(void);
extern   void     init_var(void);
extern   void     cast_skin(void);
extern   void     virus_exchange(void);
extern   int      latent_period(void);
extern   short    contaminated_leaf(void);
extern   short    aphid_contaminated(void);
extern   void     symptoms_development(void);
extern   void     distance_covered(void);
extern   void     reproduction(void);
extern   void     aphid_removal(void);
extern   void     plant_growth(void);
extern   void     weather(void);
extern   void     immigration(void);
extern   void     print_image(void);
extern   void     aphid_amount_checking(void);
extern   void     aphid_amount_adjusting(int,int);
extern   void     estimate_predation(void);
extern   void     date(short);
extern   void     rerun(void);
extern   void     main(int,char **);


FILE *openen(filename,s)
char *filename, *s;
/*************************************************************/
/* Functie:     Controleren op het aanwezig zijn van een te openen  */
/*              file. Kan de file door afwezigheid niet geopend     */
/*              worden dan wordt gelegenheid gegeven om een andere  */
/*              disk in de drive te stoppen waar de file wel op     */
/*              staat.                                              */
/*************************************************************/
 {FILE *q;
  if ((q = fopen(filename,s)) == NULL)
   {cursor(19,8);
    printf("File %s kan niet geopend worden.\n",filename);
    cursor(20,8); printf("Stop de goede disc in de goede drive.\n");
    cursor(21,8); printf("Tik daarna op een willekeurige toets.\n");
    getch();
    openen(filename,s);
   }else
  return q;
 }
```

```
int uniform(max)
int max;
/*****************************************************************/
/* Functie:     Trekken van een getal tussen 0 en het opgegeven  */
/*              maximum met een uniforme verdeling.              */
/*****************************************************************/
 {register int random;
  float tmp;
  random = rand();
  tmp = random/32767.0;
  random = max * tmp;
  return random;
 }


char *scan_input(type,row,col,max_char,error,point)
char *type;
short row, col, max_char, error, *point;
/*****************************************************************/
/* Functie:     Inlezen van karakters. Indien er een float getal */
/*              ingelezen moet worden, moet de functie met een "F"*/
/*              aangeroepen worden. Voor een integer met "I" en voor*/
/*              een letterteken met "C". Als functiewaarde wordt een*/
/*              pointer naar een string met de ingetoetste tekens */
/*              opgeleverd.                                      */
/*****************************************************************/
 {int ch;
  short i;
  char *result_ptr, result[MAX_CHAR];
  *point = 0;
  result_ptr = result;
  for (i = 1; i <= MAX_CHAR; i ++)
   {cursor(row,col+i-1);
    if (i > max_char) break;
    ch = getch();
    if (error)
     {cursor(19,7); printf(EMPTY_STRING);
      cursor(20,7); printf(EMPTY_STRING);
      cursor(21,7); printf(EMPTY_STRING);
      cursor(row,col+i-1);
      error = 0;
     }
    if (isdigit(ch))
     {if (strcmp(type,"C") == 0)
       {cursor(19,8); printf("Er moet een letter ingetypt worden.");
        cursor(20,8); printf("Hier geen cijfer toegestaan.");
        cursor(21,8);
        printf("Antwoorden met een Y (yes) of een N (no)!");
        error = 1;
        i --;
       }else
      if (strcmp(type,"S") == 0 && i == 1)
       {cursor(19,8);
       printf("Als eerste karakter moet een letter ingetypt worden.");
        cursor(20,8); printf("Hier geen cijfer toegestaan.");
        error = 1;
        i --;
       }else
```

```
    {putchar(ch);
     sprintf(result_ptr,"%d",ch-48);
     result_ptr++;
    }
  }else
 if (isalpha(ch))
  {if (strcmp(type,"C") == 0 &&
      (ch == 'n' || ch == 'N' || ch == 'y' || ch == 'Y'))
    {putchar(ch);
     if (ch == 'n' || ch == 'N')
        *result_ptr++ = 'n';
     else *result_ptr++ = 'y';
     break;
    }else
   if (strcmp(type,"S") == 0)
    {putchar(ch);
     *result_ptr++ = ch;
    }else
   if (strcmp(type,"I") == 0 || strcmp(type,"F") == 0)
    {cursor(19,8); printf("Er moet een cijfer ingetypt worden.");
     cursor(20,8); printf("Geen letters toegestaan.");
     error = 1;
     i --;
    }else
    {cursor(20,8); printf("Antwoorden met Y (yes) of N (no)!");
     error = 1;
     i --;
    }
  }else
 if (ch == '.')
  {if (strcmp(type,"F") == 0 && !*point)
    {putchar(ch);
     *result_ptr++ = '.';
     *point = 1;
    }else
    {cursor(20,8); printf("Geen punt toegestaan hier.");
     error = 1;
     i --;
    }
  }else
 if (ch == '-')
  {cursor(20,8); printf("Geen minteken toegestaan.");
   error = 1;
   i --;
  }else
 if (ch == 8)
  {cursor(row,col);
   printf("        ");
   result_ptr = result;
   *point = 0;
   i = 0;
  }else
 if (ch == 13)
  {if (strcmp(type,"C") == 0)
    {cursor(20,8); printf("Antwoorden met Y (yes) of N (no)!");
     error = 1;
     i --;
    }else break;
```

```
      }else
      {cursor(20,8); printf("Onherkenbaar teken.");
       error = 1;
       i --;
      }
    }
   *result_ptr = '\0';
   return result;
  }


float value(result,point)
char *result;
short point;
/*****************************************************************/
/* Functie:     Bepalen van de waarde van de string result.      */
/*              Opleveren van deze waarde als float.             */
/*****************************************************************/
 {char *result_ptr;
  int ch;
  float waarde = 0.0;
  short i, lengte, deler = 0;
  result_ptr = result;
  lengte = strlen(result);
  if (point) lengte -= 2;
  else lengte --;
  for (i = lengte; i >= 0; i --)
   {ch = *result_ptr++;
    if (ch == '.')
     {ch = *result_ptr++;
      deler = i + 1;
     }
    waarde += (ch-48) * pow((double)10,(double)i);
   }
  if (deler == 0) return waarde;
  else return (waarde / pow((double)10,(double)deler));
 }


screen()
 {clrscr();
  box(WIDE-2,22,1,3);
  box(WIDE-8,4,18,6);
  sascr(BOLD);
  center(1,"[ CONSTANTEN ]");
  sascr(OFF);
  center(18,"[ foutmeldingen ]");
  cursor(10,5); printf("Wilt u veranderingen aanbrengen in de
constanten? ");
 }


screen0()
 {clrscr();
  box(WIDE-2,22,1,3);
  box(WIDE-8,4,18,6);
  sascr(BOLD);
  center(1,"[ CONSTANTEN VERANDEREN ]");
```

```
   sascr(OFF);
   center(18,"[ foutmeldingen ]");
   cursor(3,5); printf("Maximum leeftijd van de luizen (dagen): %d",
   MAX_AGE);
   cursor(4,5); printf("Retentie periode van het virus (uren): %d",
   RETENTION_TIME);
   cursor(5,5); printf("Zuigtijd om het virus op te nemen (uren): %d",
   ACQUISITION_FEEDING_TIME);
   cursor(6,5); printf("Zuigtijd om het virus af te geven (uren): %d",
   TRANSMISSION_FEEDING_TIME);
   cursor(7,5); printf("Minimum vliegtemperatuur (graden C): %d",
   MIN_FLYING_TEMP);
   cursor(8,5); printf("Maximum vliegtemperatuur (graden C): %d",
   MAX_FLYING_TEMP);
   cursor(9,5); printf("Minimum looptemperatuur (graden C): %d",
   MIN_MOBILITY_TEMP);
   cursor(10,5); printf("Optimum looptemperatuur (graden C): %d",
   OPTIMUM_MOBILITY_TEMP);
   cursor(11,5); printf("Maximum looptemperatuur (graden C): %d",
   MAX_MOBILITY_TEMP);
   cursor(12,5);
   printf("Minimum temperatuur voor bladverschijning (graden C): %d",
   MIN_LEAF_APPEARANCE_TEMP);
   cursor(13,5);
   printf("Minumum temperatuur voor bladgroei (graden C): %d",
   MIN_LEAF_EXPANSION_TEMP);
   cursor(14,5);
   printf("Optimum temperatuur voor bladgroei (graden C): %d",
   OPTIMUM_GROWTH_TEMP);
   cursor(15,5);
   printf("Maximum temperatuur voor bladgroei (graden C): %d",
   MAX_GROWTH_TEMP);
   cursor(16,5);
   printf("Leeftijd waarbij maximale loopsnelheid (uren): %d",
   OPTIMUM_MOBILITY_AGE);
   cursor(17,5);
   printf("Maximale loopsnelheid van de luis (cm/uur): %d",
   MAX_VELOCITY);
  }


screen1()
 {clrscr();
  box(WIDE-2,22,1,3);
  box(WIDE-8,4,18,6);
  sascr(BOLD);
  center(1,"[ HET BIETENVELD ]");
  sascr(OFF);
  center(18,"[ foutmeldingen ]");
  center(2,"Dit is een simulatie van de verspreiding van BYV door");
  center(3,"bladluizen in een bietenveld.");
  cursor(5,5); printf("Om de hoeveel dagen wilt u een print van de
  situatie hebben?: %d",print_interval_time/24);
  cursor(6,5); printf("Geef een beginwaarde (zaadje) voor de uniforme
  verdeling: ");
  cursor(8,5); printf("De rijen van het bietenveld liggen evenwijdig
  aan de x_as.");
  cursor(9,5); printf("De lengte en de breedte van het bietenveld
```

```
   moeten in gehele");
   cursor(10,5); printf("meters opgegeven worden.");
   cursor(11,5); printf("De lengte van het veld correspondeert met de
   x_as.");
   cursor(12,5); printf("De breedte van het veld correspondeert met de
   y_as.");
   cursor(13,5); printf("Wat is de lengte van het bietenveld (x_as)?:
   %d",fieldlength);
   cursor(14,5); printf("Wat is de breedte van het bietenveld (y_as)?:
   %d",fieldwidth);
   cursor(15,5); printf("Hoeveel planten staan er per meter in de
   lengte (x_as)?: %f",lengthplants);
   cursor(16,5); printf("Hoeveel planten staan er per meter in de
   breedte (y_as)?: %f",widthplants);
  }


screen2()
 {clrscr();
  box(WIDE-2,22,1,3);
  box(WIDE-8,4,18,6);
  sascr(BOLD);
  center(1,"[ ALGEMEEN ]");
  sascr(OFF);
  center(18,"[ foutmeldingen ]");
  cursor(4,5); printf("Uitzetten luizen - infecteren planten (dagen):
  %d",interval);
  cursor(5,5); printf("Wat is de zaaidatum van de bietenplanten?");
  cursor(6,5); printf("maand (in cijfers): %d",sowing_month);
  cursor(7,5); printf("dag: %d",sowing_day);
  cursor(9,5); printf("Wat is de startdatum van de simulatie?");
  cursor(10,5); printf("maand (in cijfers): %d",month);
  cursor(11,5); printf("dag: %d",day);
  cursor(13,5); printf("Wilt u een of meerdere planten infecteren
  (Y/N)?: ");
  cursor(14,5); printf("Wilt u zelf luizen uitzetten (Y/N)?: ");
  cursor(16,5); printf("Hoe lang moet de simulatie duren (in dagen)?
  %d",maxtime/24);
 }


screen3()
 {clrscr();
  box(WIDE-2,22,1,3);
  box(WIDE-8,4,18,6);
  sascr(BOLD);
  center(1,"[ LUIZENVERDELINGEN ]");
  sascr(OFF);
  center(18,"[ foutmeldingen ]");
  if (fieldlength != 12 || fieldwidth != 12)
   {mascr(BOLD,BLINK);
    center(2,"Om van de luizenverdelingen gebruik te mogen maken moet
    het veld");
    center(3,"12 bij 12 meter zijn!");
    sascr(OFF);
   }
  cursor(8,5); printf("Welke leeftijd hebben de luizen bij het
  uitzetten (in dagen)?: ");
```

```
   cursor(9,5); printf("Welke luizenverdeling wilt u gebruiken?");
   cursor(10,5); printf("Eigen, vroeg 65, vroeg 9, vroeg 2, laat 65,
   laat 9 of laat 2.");
   cursor(11,5); printf("  0       1         2        3        4
   5   of    6  : ");
  }


screen4()
 {clrscr();
  box(WIDE-2,22,1,3);
  box(WIDE-8,4,18,6);
  sascr(BOLD);
  center(1,"[ ZELF LUIZEN UITZETTEN ]");
  sascr(OFF);
  center(18,"[ foutmeldingen ]");
  cursor(5,5); printf("Hoeveel luizen wilt u uitzetten?:");
  cursor(6,5); printf("Welke leeftijd (in dagen) hebben ze?:");
  cursor(7,5); printf("Zijn ze besmet met BYV (Y/N)?:");
  cursor(8,5); printf("Zijn ze gevleugeld (Y/N)?:");
  cursor(9,5); printf("Wat zijn de coordinaten in het veld?");
  cursor(10,5); printf("Geef een waarde voor de x_as (in planten):");
  cursor(11,5); printf("Geef een waarde voor de y_as (in planten):");
  cursor(12,5); printf("Wilt u nog meer luizen uitzetten (Y/N)?:");
 }


screen5()
 {clrscr();
  box(WIDE-2,22,1,3);
  box(WIDE-8,4,18,6);
  sascr(BOLD);
  center(1,"[ PLANTEN INFECTEREN ]");
  sascr(OFF);
  center(18,"[ foutmeldingen ]");
  cursor(8,5); printf("Wat zijn de coordinaten in het veld?");
  cursor(9,5); printf("Geef een waarde voor de x_as (in planten):");
  cursor(10,5); printf("Geef een waarde voor de y_as (in planten):");
  cursor(11,5); printf("Wilt u nog meer planten infecteren (Y/N)?:");
 }


screen6()
 {int i;
  char info[30];
  clrscr();
  strcpy(info,"Computer niet uitzetten!");
  mascr(2,BOLD,BLINK);
  for (i = 0; i < 24; i ++)
   {cursor(i+1,70);
    printf("%c",info[i]);
   }
  sascr(OFF);
 }


screen7()
 {clrscr();
```

```
  box(WIDE-2,22,1,3);
  box(WIDE-8,4,18,6);
  sascr(BOLD);
  center(1,"[ PARMFILE ]");
  sascr(OFF);
  center(18,"[ foutmeldingen ]");
  cursor(10,5); printf("Wilt u een parmfile maken (Y/N)?: ");
 }


screen8()
 {clrscr();
  box(WIDE-2,22,1,3);
  box(WIDE-8,4,18,6);
  sascr(BOLD);
  center(1,"[ OUTPUTFILES ]");
  sascr(OFF);
  center(18,"[ foutmeldingen ]");
  cursor(8,5); printf("Wilt u een file met alle simulatiegegevens? ");
  cursor(9,5); printf("Wilt u de gegevens in een plattegrond van 't
  veld? ");
  cursor(10,5); printf("Wilt u de luizenaantallen in een grafiek op
  file? ");
  cursor(11,5); printf("Wilt u de vergeelde planten in een grafiek op
  file? ");
 }


create_aphid(age,contaminated,x,y,winged)
int age, x, y;
short contaminated, winged;
/*****************************************************************/
/* Functie:     Creeren van luizen met de eigenschappen welke in het*/
/*              interface van de functie meegegeven zijn.          */
/*              De nieuw gecreeerde luis wordt vooraan de dubble   */
/*              linked list, waar alle luizen in opgenomen zijn,   */
/*              geplaatst.                                         */
/*****************************************************************/
 {struct aphid *next_aphid;
  if (first_aphid == NULL)
   {first_aphid = (struct aphid *) calloc(1,sizeof(struct aphid));
    last_aphid = first_aphid;
    first_aphid->next = NULL;
    first_aphid->pred = NULL;
   }else
   {next_aphid = (struct aphid *) calloc(1,sizeof(struct aphid));
    first_aphid->pred = next_aphid;
    next_aphid->next = first_aphid;
    next_aphid->pred = NULL;
    first_aphid = next_aphid;
   }
  first_aphid->time_of_birth = time - age;
  first_aphid->winged = winged;
  first_aphid->x = x;
  first_aphid->y = y;
  if (contaminated)
     first_aphid->time_of_contamination = time;
  else first_aphid->time_of_contamination = time - RETENTION_TIME;
```

```
   field[x/lengthfraction][y/widthfraction]->nbr_of_aphids ++;
   tot_aphids ++;
  }


delete_aphid()
/*****************************************************************/
/* Functie:   Verwijderen van luizen uit de double linked list.  */
/*            Herstellen van de structuur van de double linked list.*/
/*****************************************************************/
 {int x, y;
  struct aphid *dead_aphid;
  x = this_aphid->x/lengthfraction;
  y = this_aphid->y/widthfraction;
  dead_aphid = this_aphid;
  if (this_aphid->next != NULL &&
      this_aphid->pred != NULL)
   {this_aphid->next->pred = this_aphid->pred;
    this_aphid->pred->next = this_aphid->next;
    this_aphid = this_aphid->pred;
   }else
  if (this_aphid->next == NULL &&
      this_aphid->pred != NULL)
   {this_aphid->pred->next = NULL;
    last_aphid = this_aphid->pred;
    this_aphid = last_aphid;
   }else
  if (this_aphid->next != NULL &&
      this_aphid->pred == NULL)
   {this_aphid->next->pred = NULL;
    first_aphid = this_aphid->next;
    dummy_aphid->next = first_aphid;
    this_aphid = dummy_aphid;
   }else
   {dummy_aphid->next = NULL;
    this_aphid = dummy_aphid;
    first_aphid = last_aphid = NULL;
   }
  free(dead_aphid);
  field[x][y]->nbr_of_aphids --;
  tot_aphids --;
 }


init_dialog(inputfile)
char *inputfile;
/*****************************************************************/
/* Functie: Inlezen van alle beginwaarden voor de simulatie die  */
/*          door de gebruiker opgegeven kunnen worden. In deze   */
/*          functie vindt dus de dialoog met de gebruiker van het*/
/*          simulatieprogramma plaats.                           */
/*****************************************************************/
 {short point, error = 0;
  int maxdays = 0;
  int aphid_distribution = 0;
  int i, j;
  float answer;
  char choice[2];
```

```
char result[MAX_CHAR];
char filename[16];
FILE *q, *in;
/*                                                                    */
/* Om gebruik te kunnen maken van diverse schermcommando's:           */
_colorflag = 1;
/*                                                                    */
/* Hier worden de namen en het aantal dagen van de verschillende      */
/* maanden geinitialiseerd. Deze gegevens zijn al nodig in deze       */
/* functie.                                                           */
monthdays[1] = 31;  monthdays[2] = 28;  monthdays[3] = 31;
monthdays[4] = 30;  monthdays[5] = 31;  monthdays[6] = 30;
monthdays[7] = 31;  monthdays[12] = 31; monthdays[9] = 30;
monthdays[10] = 31; monthdays[11] = 30; monthdays[8] = 31;
monthname[1] = "januari";          monthname[2] = "februari";
monthname[3] = "maart";            monthname[4] = "april";
monthname[5] = "mei";              monthname[6] = "juni";
monthname[7] = "juli";             monthname[8] = "augustus";
monthname[9] = "september";        monthname[10] = "oktober";
monthname[11] = "november";        monthname[12] = "december";
/*                                                                    */
/* Vragen of de gebruiker een parmfile wil maken van de in te         */
/* voeren gegevens.                                                   */
if (!simulation_data_from_file)
 {screen7();
  strcpy(choice,scan_input("C",10,38,1,0,&point));
  if (strcmp(choice,"y") == 0 || strcmp(choice,"Y") == 0)
   {cursor(11,5); printf("Geef een naam voor de parmfile (max 8
    karakters): ");
    LOOP /* screen7 */
     {strcpy(result,scan_input("S",11,55,MAX_CHAR,error,&point));
      if (strlen(result))
       {strcpy(parmfilename,result);
        strcat(parmfilename,".par");
       }else
       {novalue(); error = 1;
        continue;
       }
      if (terminate()) break;
      else continue;
     } /* end screen7 */
    par = openen(parmfilename,"w");
    parmfile = 1;
   }else parmfile = 0;
 }
/*                                                                    */
/* Lezen van de opstartwaarden van de laatst uitgevoerde simulatie*/
if (!parmfile)
 {q = fopen("constant.dat","r");
  fscanf(q,"%d",&MAX_AGE);
  fscanf(q,"%d",&RETENTION_TIME);
  fscanf(q,"%d",&ACQUISITION_FEEDING_TIME);
  fscanf(q,"%d",&TRANSMISSION_FEEDING_TIME);
  fscanf(q,"%d",&MIN_FLYING_TEMP);
  fscanf(q,"%d",&MAX_FLYING_TEMP);
  fscanf(q,"%d",&MIN_MOBILITY_TEMP);
  fscanf(q,"%d",&OPTIMUM_MOBILITY_TEMP);
  fscanf(q,"%d",&MAX_MOBILITY_TEMP);
```

```
   fscanf(q,"%d",&MIN_LEAF_APPEARANCE_TEMP);
   fscanf(q,"%d",&MIN_LEAF_EXPANSION_TEMP);
   fscanf(q,"%d",&OPTIMUM_GROWTH_TEMP);
   fscanf(q,"%d",&MAX_GROWTH_TEMP);
   fscanf(q,"%d",&OPTIMUM_MOBILITY_AGE);
   fscanf(q,"%d",&MAX_VELOCITY);
   fclose(q);
  }
/*                                                                    */
/* Indien niet alle gegevens van file gehaald moeten worden worden*/
/* de opstartwaarden van de laatste simulatie van file ingelezen. */
if (!simulation_data_from_file && !parmfile)
 {q = fopen("byv.dat","r");
  fscanf(q,"%d",&print_data);
  fscanf(q,"%d",&print_picture);
  fscanf(q,"%d",&print_luizengrafiek);
  fscanf(q,"%d",&print_plantengrafiek);
  fscanf(q,"%d",&print_interval_time);
  fscanf(q,"%d",&fieldlength);
  fscanf(q,"%d",&fieldwidth);
  fscanf(q,"%f",&lengthplants);
  fscanf(q,"%f",&widthplants);
  fscanf(q,"%d",&sowing_month);
  fscanf(q,"%d",&sowing_day);
  fscanf(q,"%d",&month);
  fscanf(q,"%d",&day);
  fscanf(q,"%d",&maxtime);
  fclose(q);
 }
/*                                                                    */
/* Indien alle gegevens van file moeten komen, wordt die file hier*/
/* geopend.                                                           */
if (simulation_data_from_file && !parmfile)
 {strcpy(filename,inputfile);
  strcat(filename,".par");
  in = openen(filename,"r");
 }
/*                                                                    */
/* Vragen naar wensen omtrent de outputfiles en vragen of de          */
/* diverse constanten veranderd moeten worden.                        */
if (!simulation_data_from_file) screen8();
if (simulation_data_from_file)
   fscanf(in,"%s",choice);
else strcpy(choice,scan_input("C",8,49,1,0,&point));
if (parmfile) fprintf(par,"%s\n",choice);
if (strcmp(choice,"y") == 0 || strcmp(choice,"Y") == 0)
   print_data = 1;
else print_data = 0;
if (simulation_data_from_file)
   fscanf(in,"%s",choice);
else strcpy(choice,scan_input("C",9,56,1,0,&point));
if (parmfile) fprintf(par,"%s\n",choice);
if (strcmp(choice,"y") == 0 || strcmp(choice,"Y") == 0)
   print_picture = 1;
else print_picture = 0;
if (simulation_data_from_file)
   fscanf(in,"%s",choice);
else strcpy(choice,scan_input("C",10,55,1,0,&point));
```

```
if (parmfile) fprintf(par,"%s\n",choice);
if (strcmp(choice,"y") == 0 || strcmp(choice,"Y") == 0)
   print_luizengrafiek = 1;
else print_luizengrafiek = 0;
if (simulation_data_from_file)
   fscanf(in,"%s",choice);
else strcpy(choice,scan_input("C",11,57,1,0,&point));
if (parmfile) fprintf(par,"%s\n",choice);
if (strcmp(choice,"y") == 0 || strcmp(choice,"Y") == 0)
   print_plantengrafiek = 1;
else print_plantengrafiek = 0;
if (!simulation_data_from_file) screen();
if (simulation_data_from_file)
   fscanf(in,"%s",choice);
else strcpy(choice,scan_input("C",10,55,1,0,&point));
if (parmfile) fprintf(par,"%s\n",choice);
if (strcmp(choice,"y") == 0 || strcmp(choice,"Y") == 0)
   change_constants(in);
/*                                                                  */
/* Het inlezen van de gewenste gegevens kan nu beginnen. Indien     */
/* simulation_data_from_file 1 is, worden alle gegevens van file    */
/* ingelezen.                                                       */
if (!simulation_data_from_file) screen1();
LOOP{/* screen1 */
if (simulation_data_from_file)
 {fscanf(in,"%d",&print_interval_time);
  print_interval_time *= 24;
 }else LOOP
 {strcpy(result,scan_input("I",5,67,3,error,&point));
  if (strlen(result))
   {answer = value(result,point);
    if (answer == 0)
     {cursor(20,8); printf("Nul dagen is niet toegestaan!");
      cursor(5,67); printf("        "); error = 1;
      continue;
     }
    print_interval_time = answer * 24;
    if (parmfile) fprintf(par,"%d\n",print_interval_time/24);
    break;
   }else
  if (print_interval_time == 0)
   {novalue(); error = 1;
    continue;
   }else break;
 }
if (simulation_data_from_file)
   fscanf(in,"%d",&seed);
else LOOP
 {strcpy(result,scan_input("I",6,63,3,error,&point));
  if (strlen(result))
   {answer = value(result,point);
    seed = answer;
    seed = abs(seed);
    if (parmfile) fprintf(par,"%d\n",seed);
    break;
   }else
  if (seed == 0)
   {novalue(); error = 1;
```

```
     continue;
    }else break;
  }
 if (simulation_data_from_file)
    fscanf(in,"%d",&fieldlength);
 else LOOP
  {strcpy(result,scan_input("I",13,50,2,error,&point));
   if (strlen(result))
    {answer = value(result,point);
     if (answer > MAXLENGTH)
      {cursor(20,8); printf("De lengte mag niet groter dan %d
       zijn!",MAXLENGTH);
       cursor(13,50); printf("        "); error = 1;
       continue;
      }
     fieldlength = answer;
     if (parmfile) fprintf(par,"%d\n",fieldlength);
     break;
    }else
   if (fieldlength == 0)
    {novalue(); error = 1;
     continue;
    }else break;
  }
 if (simulation_data_from_file)
    fscanf(in,"%d",&fieldwidth);
 else LOOP
  {strcpy(result,scan_input("I",14,51,2,error,&point));
   if (strlen(result))
    {answer = value(result,point);
     if (answer > MAXWIDTH)
      {cursor(20,8); printf("De breedte mag niet groter dan %d
       zijn!",MAXWIDTH);
       cursor(14,51); printf("        "); error = 1;
       continue;
      }
     fieldwidth = answer;
     if (parmfile) fprintf(par,"%d\n",fieldwidth);
     break;
    }else
   if (fieldwidth == 0)
    {novalue(); error = 1;
     continue;
    }else break;
  }
 if (simulation_data_from_file)
    fscanf(in,"%f",&lengthplants);
 else LOOP
  {strcpy(result,scan_input("F",15,62,8,error,&point));
   if (strlen(result))
    {answer = value(result,point);
     if (answer < 0 || answer > MAXLENGTH/fieldlength)
      {cursor(19,8); printf("Er mogen ten hoogste %d planten per
       meter in de lengte staan.",
       MAXLENGTH/fieldlength);
       cursor(20,8); printf("Er moet ten minste 1 plant per meter in
       de lengte staan.");
       cursor(15,62); printf("        "); error = 1;
```

```
      continue;
     }
    lengthplants = answer;
    if (parmfile) fprintf(par,"%f\n",lengthplants);
    break;
   }else
  if (lengthplants == 0)
   {novalue(); error = 1;
    continue;
   }else break;
 }
if (simulation_data_from_file)
   fscanf(in,"%f",&widthplants);
else LOOP
 {strcpy(result,scan_input("F",16,63,8,error,&point));
  if (strlen(result))
   {answer = value(result,point);
    if (answer < 0 || answer > MAXWIDTH/fieldwidth)
     {cursor(19,8); printf("Er mogen ten hoogste %d planten per
      meter in de breedte staan.",
      MAXWIDTH/fieldwidth);
      cursor(20,8); printf("Er moet ten minste 1 plant per meter in
      de breedte staan.");
      cursor(16,63); printf("        "); error = 1;
      continue;
     }
    widthplants = answer;
    if (parmfile) fprintf(par,"%f\n",widthplants);
    break;
   }else
  if (widthplants == 0)
   {novalue(); error = 1;
    continue;
   }else break;
 }
if (terminate()) break;
else continue;
}/* end screen1 */
if (print_data)
 {fprintf(p1,"De lengte van het bietenveld is %d meter.\n",
  fieldlength);
  fprintf(p1,"De breedte van het bietenveld is %d meter.\n",
  fieldwidth);
  fprintf(p1,"Er staan %3.1f planten per meter in de lengte.\n",
  lengthplants);
  fprintf(p1,"Er staan %3.1f planten per meter in de breedte.\n",
  widthplants);
 }
tot_widthplants = widthplants * fieldwidth;
tot_lengthplants = lengthplants * fieldlength;
tot_plants = tot_widthplants * tot_lengthplants;
widthfraction = 100/widthplants;
lengthfraction = 100/lengthplants;
if (print_data)
 {fprintf(p1,"\nIn de lengte staan %d planten.\n",tot_lengthplants);
  fprintf(p1,"In de breedte staan %d planten.\n",tot_widthplants);
  fprintf(p1,"In totaal staan er %d planten in het veld.\n\n",
  tot_plants);
```

```
 }
for (i = 0; i < tot_lengthplants; i ++)
for (j = 0; j < tot_widthplants; j ++)
 {field[i][j] = (struct beet *) calloc(1,sizeof(struct beet));
  field[i][j]->time_of_contamination = -1;
  field[i][j]->healthy_leaves = MAX_LEAVES;
 }
if (!simulation_data_from_file) screen2();
LOOP{/* screen2 */
if (simulation_data_from_file)
   fscanf(in,"%d",&interval);
else LOOP
 {strcpy(result,scan_input("I",4,52,2,error,&point));
  if (strlen(result))
   {answer = value(result,point);
    interval = answer;
    if (parmfile) fprintf(par,"%d\n",interval);
    break;
   }else
  if (interval == 0)
   {novalue(); error = 1;
    continue;
   }else break;
 }
if (simulation_data_from_file)
   fscanf(in,"%d",&sowing_month);
else LOOP
 {strcpy(result,scan_input("I",6,25,1,error,&point));
  if (strlen(result))
   {answer = value(result,point);
    if (answer < 4 || answer > 10)
     {cursor(20,8); printf("De zaaidatum moet binnen de maanden
      april t/m oktober liggen!");
      cursor(6,25); printf("        "); error = 1;
      continue;
     }
    sowing_month = answer;
    if (parmfile) fprintf(par,"%d\n",sowing_month);
    break;
   }else
  if (sowing_month == 0)
   {novalue(); error = 1;
    continue;
   }else break;
 }
if (simulation_data_from_file)
   fscanf(in,"%d",&sowing_day);
else LOOP
 {strcpy(result,scan_input("I",7,10,2,error,&point));
  if (strlen(result))
   {answer = value(result,point);
    if (answer < 0 || answer > monthdays[sowing_month])
     {cursor(20,8); printf("De waarde moet binnen 0 t/m %d liggen!",
      monthdays[sowing_month]);
      cursor(7,10); printf("        "); error = 1;
      continue;
     }
    sowing_day = answer;
```

```
     if (parmfile) fprintf(par,"%d\n",sowing_day);
     break;
    }else
   if (sowing_day == 0)
    {novalue(); error = 1;
     continue;
    }else break;
  }
if (simulation_data_from_file)
   fscanf(in,"%d",&month);
else LOOP
 {strcpy(result,scan_input("I",10,25,1,error,&point));
  if (strlen(result))
   {answer = value(result,point);
    if (answer < sowing_month || answer > 10)
     {cursor(20,8); printf("De startdatum moet binnen de maanden %s
      t/m oktober liggen!",
      monthname[sowing_month]);
      cursor(10,25); printf("        "); error = 1;
      continue;
     }
    month = answer;
    if (parmfile) fprintf(par,"%d\n",month);
    break;
   }else
  if (month == 0)
   {novalue(); error = 1;
    continue;
   }else break;
  }
if (simulation_data_from_file)
   fscanf(in,"%d",&day);
else LOOP
 {strcpy(result,scan_input("I",11,10,2,error,&point));
  if (strlen(result))
   {answer = value(result,point);
    if (answer <= sowing_day && sowing_month == month)
     {cursor(20,8); printf("Dag moet groter zijn dan %d (dag van
      zaaien).",sowing_day);
      cursor(11,10); printf("        "); error = 1;
      continue;
     }
    if (answer < 0 || answer > monthdays[month])
     {cursor(20,8); printf("De waarde moet binnen 0 t/m %d liggen!",
      monthdays[month]);
      cursor(11,10); printf("        "); error = 1;
      continue;
     }
    day = answer;
    if (parmfile) fprintf(par,"%d\n",day);
    break;
   }else
  if (day == 0)
   {novalue(); error = 1;
    continue;
   }else break;
  }
if (terminate()) break;
```

```
else continue;
}/* end screen2 */
if (print_data)
 {fprintf(p1,"De bieten zijn gezaaid op %d %s.\n",
  sowing_day,monthname[sowing_month]);
  fprintf(p1,"De simulatie is gestart op %d %s.\n",
  day,monthname[month]);
 }
/*                                                              */
/* Nu bekend is wanneer de bieten gezaaid zijn en wanneer de    */
/* simulatie begint, moet init_var() aangeroepen worden.        */
if (!parmfile)
 {clrscr();
  mascr(2,BOLD,BLINK);
  center(12,"even geduld a.u.b.");
  sascr(OFF);
  init_var();
 }
if (!simulation_data_from_file) screen2();
/*                                                              */
/* Na het aanroepen van init_var() kunnen de planten geinfecteerd */
/* en de luizen uitgezet worden.                                */
if (simulation_data_from_file)
   fscanf(in,"%s",choice);
else strcpy(choice,scan_input("C",13,55,1,error,&point));
if (parmfile) fprintf(par,"%s\n",choice);
if (strcmp(choice,"y") == 0 || strcmp(choice,"Y") == 0)
 {contaminate_plant(in);
  if (!simulation_data_from_file) screen2();
 }
if (simulation_data_from_file)
   fscanf(in,"%s",choice);
else strcpy(choice,scan_input("C",14,42,1,error,&point));
if (parmfile) fprintf(par,"%s\n",choice);
if (strcmp(choice,"y") == 0 || strcmp(choice,"Y") == 0)
 {aphid_data_from_file = 0;
  put_aphids(in);
  if (!simulation_data_from_file) screen2();
 }else
 {aphid_data_from_file = 1;
  if (simulation_data_from_file)
   {fscanf(in,"%d",&initial_age);
    initial_age *= 24;
   }else
   {screen3();
    LOOP
     {strcpy(result,scan_input("I",8,68,2,error,&point));
      if (strlen(result))
       {answer = value(result,point);
        if (answer < 0 || answer > 42)
         {cursor(20,8); printf("Luizen worden in dit model niet
          ouder dan 42 dagen.");
          cursor(8,57); printf("        "); error = 1;
          continue;
         }
        initial_age = answer;
        if (parmfile) fprintf(par,"%d\n",initial_age);
        initial_age *= 24;
```

```
          break;
         }else
        if (initial_age == 0)
         {novalue(); error = 1;
          continue;
         }else break;
       }
     }
    if (simulation_data_from_file)
       fscanf(in,"%d",&aphid_distribution);
    else LOOP
     {strcpy(result,scan_input("I",11,67,1,error,&point));
      if (strlen(result))
       {answer = value(result,point);
        if (answer < 0 || answer > 6)
         {cursor(20,8); printf("U moet kiezen uit 0,1,2,3,4,5,6!!");
          cursor(11,67); printf("        "); error = 1;
          continue;
         }
        aphid_distribution = answer;
        if (parmfile) fprintf(par,"%d\n",aphid_distribution);
        break;
       }else
      if (aphid_distribution == 0)
       {novalue(); error = 1;
        continue;
       }else break;
     }
    if (print_data)
     {sprintf(filename,"luizen%d.dat",aphid_distribution);
      fprintf(p1,"File %s is gebruikt voor de luizenaantallen.\n",
      filename);
      fprintf(p1,"Leeftijd luizen op moment van uitzetten: %d
      dagen.\n",initial_age/24);
     }
    if (!parmfile)
     {q = openen(filename,"r");
      for (i = 0; i < 214; i ++)
      fscanf(q,"%4d",(field_counting + i));
      fclose(q);
     }
    if (!simulation_data_from_file) screen2();
   }
  if (simulation_data_from_file)
   {fscanf(in,"%d",&maxtime);
    maxtime *= 24;
   }else LOOP
   {strcpy(result,scan_input("I",16,50,3,error,&point));
    if (strlen(result))
     {answer = value(result,point);
      for (i = month; i <= 10; i ++)
      maxdays += monthdays[i];
      maxdays -= day;
      if (answer > maxdays)
       {cursor(20,8); printf("Er kunnen maximaal %d dagen gesimuleerd
        worden vanaf %d %s!",
        maxdays,day,monthname[month]);
        maxdays = 0;
```

```
      cursor(16,50); printf("        "); error = 1;
      continue;
     }
    maxtime = answer;
    if (parmfile) fprintf(par,"%d\n",maxtime);
    maxtime *= 24;
    break;
   }else
  if (maxtime == 0)
   {novalue(); error = 1;
    continue;
   }else break;
 }
/*                                                                    */
/* Sluiten van de parmfile indien geopend.                            */
if (parmfile)
   fclose(par);
else
 {screen6();
  if (print_data) fprintf(p1,"De simulatie zal %d dagen duren.\n",
   maxtime/24);
  if (simulation_data_from_file) fclose(in);
 }
/*                                                                    */
/* Wegschrijven van de opstartwaarden van deze run.                   */
q = fopen("constant.dat","w");
fprintf(q,"%d\n",MAX_AGE);
fprintf(q,"%d\n",RETENTION_TIME);
fprintf(q,"%d\n",ACQUISITION_FEEDING_TIME);
fprintf(q,"%d\n",TRANSMISSION_FEEDING_TIME);
fprintf(q,"%d\n",MIN_FLYING_TEMP);
fprintf(q,"%d\n",MAX_FLYING_TEMP);
fprintf(q,"%d\n",MIN_MOBILITY_TEMP);
fprintf(q,"%d\n",OPTIMUM_MOBILITY_TEMP);
fprintf(q,"%d\n",MAX_MOBILITY_TEMP);
fprintf(q,"%d\n",MIN_LEAF_APPEARANCE_TEMP);
fprintf(q,"%d\n",MIN_LEAF_EXPANSION_TEMP);
fprintf(q,"%d\n",OPTIMUM_GROWTH_TEMP);
fprintf(q,"%d\n",MAX_GROWTH_TEMP);
fprintf(q,"%d\n",OPTIMUM_MOBILITY_AGE);
fprintf(q,"%d\n",MAX_VELOCITY);
fclose(q);
q = fopen("byv.dat","w");
fprintf(q,"%d\n",print_data);
fprintf(q,"%d\n",print_picture);
fprintf(q,"%d\n",print_luizengrafiek);
fprintf(q,"%d\n",print_plantengrafiek);
fprintf(q,"%d\n",print_interval_time);
fprintf(q,"%d\n",fieldlength);
fprintf(q,"%d\n",fieldwidth);
fprintf(q,"%f\n",lengthplants);
fprintf(q,"%f\n",widthplants);
fprintf(q,"%d\n",sowing_month);
fprintf(q,"%d\n",sowing_day);
fprintf(q,"%d\n",month);
fprintf(q,"%d\n",day);
fprintf(q,"%d\n",maxtime);
fclose(q);
```

```
  if (print_data) fprintf(p1,"\f");
  /*                                                              */
  /* Printen van de waarden van de diverse constanten.            */
  if (print_data)
   {fprintf(p1,"MAX_AGE                      %d\n",MAX_AGE);
    fprintf(p1,"RETENTION_TIME               %d\n",RETENTION_TIME);
    fprintf(p1,"ACQUISITION_FEEDING_TIME  %d\n",
    ACQUISITION_FEEDING_TIME);
    fprintf(p1,"TRANSMISSION_FEEDING_TIME %d\n",
    TRANSMISSION_FEEDING_TIME);
    fprintf(p1,"MIN_FLYING_TEMP              %d\n",MIN_FLYING_TEMP);
    fprintf(p1,"MAX_FLYING_TEMP              %d\n",MAX_FLYING_TEMP);
    fprintf(p1,"MIN_MOBILITY_TEMP            %d\n",MIN_MOBILITY_TEMP);
    fprintf(p1,"OPTIMUM_MOBILITY_TEMP     %d\n",
    OPTIMUM_MOBILITY_TEMP);
    fprintf(p1,"MAX_MOBILITY_TEMP            %d\n",MAX_MOBILITY_TEMP);
    fprintf(p1,"MIN_LEAF_APPEARANCE_TEMP  %d\n",
    MIN_LEAF_APPEARANCE_TEMP);
    fprintf(p1,"MIN_LEAF_EXPANSION_TEMP   %d\n",
    MIN_LEAF_EXPANSION_TEMP);
    fprintf(p1,"OPTIMUM_GROWTH_TEMP          %d\n",OPTIMUM_GROWTH_TEMP);
    fprintf(p1,"MAX_GROWTH_TEMP              %d\n",MAX_GROWTH_TEMP);
    fprintf(p1,"OPTIMUM_MOBILITY_AGE         %d\n",OPTIMUM_MOBILITY_AGE);
    fprintf(p1,"MAX_VELOCITY                 %d\n",MAX_VELOCITY);
    fprintf(p1,"\f");
   }
 }


contaminate_plant(in)
FILE *in;
/*****************************************************************/
/* Functie: Inlezen van de waarden nodig om een bepaalde plant in   */
/*          het bietenveld te besmetten met BYV.                    */
/*****************************************************************/
 {int x = 0, y = 0;
  float answer;
  short point, error = 0;
  char choice[2];
  char result[MAX_CHAR];
  if (print_data) fprintf(p1,"\n");
  if (!simulation_data_from_file) screen5();
  LOOP{/* screen5 */
  if (simulation_data_from_file)
   {fscanf(in,"%d",&x);
    x --;
   }else LOOP
   {strcpy(result,scan_input("I",9,47,2,error,&point));
    if (strlen(result))
     {answer = value(result,point);
      if (answer > tot_lengthplants || answer < 1)
       {cursor(20,8); printf("De waarde voor de x_as moet groter zijn
        dan 0 en kleiner dan %d!",
        tot_lengthplants);
        cursor(9,47); printf("        "); error = 1;
        continue;
       }
      x = answer;
```

```
      if (parmfile) fprintf(par,"%d\n",x);
      x --;
      break;
     }else
    if (x == 0)
     {novalue(); error = 1;
      continue;
     }else break;
   }
  if (simulation_data_from_file)
   {fscanf(in,"%d",&y);
    y --;
   }else LOOP
   {strcpy(result,scan_input("I",10,47,2,error,&point));
    if (strlen(result))
     {answer = value(result,point);
      if (answer > tot_widthplants || answer < 1)
       {cursor(20,8); printf("De waarde voor de y_as moet groter zijn
        dan 0 en kleiner dan %d!",
        tot_widthplants);
        cursor(10,47); printf("        "); error = 1;
        continue;
       }
      y = answer;
      if (parmfile) fprintf(par,"%d\n",y);
      y --;
      break;
     }else
    if (y == 0)
     {novalue(); error = 1;
      continue;
     }else break;
   }
  if (terminate()) break;
  else continue;
  }/* end screen5 */
  if (print_data) fprintf(pl,"Plant %d,%d is met het virus besmet.\n",
  x+1,y+1);
  field[x][y]->time_of_contamination = 0;
  field[x][y]->healthy_leaves = floor(nbr_of_leaves);
  if (simulation_data_from_file)
     fscanf(in,"%s",choice);
  else strcpy(choice,scan_input("C",11,47,1,error,&point));
  if (parmfile) fprintf(par,"%s\n",choice);
  if (strcmp(choice,"y") == 0 || strcmp(choice,"Y") == 0)
     contaminate_plant(in);
  else if (print_data) fprintf(pl,"\n");
 }


put_aphids(in)
FILE *in;
/*****************************************************************/
/* Functie: Inlezen van de leeftijd, het al of niet besmet zijn,    */
/*          het al of niet gevleugeld zijn en de plaats in het veld */
/*          van de luis alsmede het aantal luizen met bovengenoemde */
/*          eigenschappen dat uitgezet moet worden.                 */
/*****************************************************************/
```

```
{int i, n = 0, age = 0, x = 0, y = 0;
 short contaminated = 0, winged = 0;
 float answer;
 short point, error = 0;
 char choice[2];
 char result[MAX_CHAR];
 if (print_data) fprintf(p1,"\n");
 if (!simulation_data_from_file) screen4();
 LOOP{/* screen4 */
 if (simulation_data_from_file)
    fscanf(in,"%d",&n);
 else LOOP
  {strcpy(result,scan_input("I",5,38,3,error,&point));
   if (strlen(result))
    {answer = value(result,point);
     if (tot_aphids + answer > MAX_APHIDS)
      {cursor(20,8); printf("Er mogen niet meer dan %d luizen
       uitgezet worden!",
       MAX_APHIDS);
       cursor(5,38); printf("        "); error = 1;
       continue;
      }
     n = answer;
     if (parmfile) fprintf(par,"%d\n",n);
     break;
    }else
   if (n == 0)
    {novalue(); error = 1;
     continue;
    }else break;
  }
 if (simulation_data_from_file)
  {fscanf(in,"%d",&age);
   age *= 24;
  }else LOOP
  {strcpy(result,scan_input("I",6,42,2,error,&point));
   if (strlen(result))
    {answer = value(result,point);
     if (answer > MAX_AGE)
      {cursor(20,8); printf("Luizen worden in dit model niet ouder
       dan %d dagen!",MAX_AGE);
       cursor(6,42); printf("        "); error = 1;
       continue;
      }
     age = answer * 24;
     if (parmfile) fprintf(par,"%d\n",age);
     break;
    }else
   if (age == 0)
    {novalue(); error = 1;
     continue;
    }else break;
  }
 if (simulation_data_from_file)
    fscanf(in,"%s",choice);
 else strcpy(choice,scan_input("C",7,35,1,error,&point));
 if (parmfile) fprintf(par,"%s\n",choice);
 if (strlen(result) || simulation_data_from_file)
```

```
 {if (strcmp(choice,"y") == 0 || strcmp(choice,"Y") == 0)
     contaminated = 1;
  else contaminated = 0;
 }
if (simulation_data_from_file)
   fscanf(in,"%s",choice);
else strcpy(choice,scan_input("C",8,31,1,error,&point));
if (parmfile) fprintf(par,"%s\n",choice);
if (strlen(result) || simulation_data_from_file)
 {if (strcmp(choice,"y") == 0 || strcmp(choice,"Y") == 0)
     winged = 1;
  else winged = 0;
 }
if (simulation_data_from_file)
 {fscanf(in,"%d",&x);
  x --;
 }else LOOP
 {strcpy(result,scan_input("I",10,47,2,error,&point));
  if (strlen(result))
   {answer = value(result,point);
    if (answer > tot_lengthplants || answer < 1)
     {cursor(20,8); printf("De waarde voor de x_as moet groter zijn
      dan 0 en kleiner dan %d!",
      tot_lengthplants);
      cursor(10,47); printf("        "); error = 1;
      continue;
     }
    x = answer;
    if (parmfile) fprintf(par,"%d\n",x);
    x --;
    break;
   }else
  if (x == 0)
   {novalue(); error = 1;
    continue;
   }else break;
 }
if (simulation_data_from_file)
 {fscanf(in,"%d",&y);
  y --;
 }else LOOP
 {strcpy(result,scan_input("I",11,47,2,error,&point));
  if (strlen(result))
   {answer = value(result,point);
    if (answer > tot_widthplants || answer < 1)
     {cursor(20,8); printf("De waarde voor de y_as moet groter zijn
      dan 0 en kleiner dan %d!",
      tot_widthplants);
      cursor(11,47); printf("        "); error = 1;
      continue;
     }
    y = answer;
    if (parmfile) fprintf(par,"%d\n",y);
    y --;
    break;
   }else
  if (y == 0)
   {novalue(); error = 1;
```

```
      continue;
     }else break;
   }
  if (terminate()) break;
  else continue;
  }/* end screen4 */
  if (print_data)
   {fprintf(p1,"Er zijn %d luizen met een leeftijd van %d dagen
    uitgezet.\n",
    n,age/24);
    if (contaminated) fprintf(p1,"Ze zijn besmet met BYV.\n");
    else fprintf(p1,"Ze zijn niet besmet met BYV.\n");
    if (winged) fprintf(p1,"Ze zijn gevleugeld.\n");
    else fprintf(p1,"Ze zijn niet gevleugeld.\n");
    fprintf(p1,"Ze zijn uitgezet op plant %d,%d.\n",x+1,y+1);
   }
  /*                                                              */
  /* luizen random op de plant zetten.                            */
  for (i = 1; i <= n; i ++)
   {int x_plant, y_plant;
    x_plant = x * lengthfraction + uniform(lengthfraction);
    y_plant = y * widthfraction + uniform(widthfraction);
    create_aphid(age,contaminated,x_plant,y_plant,winged);
   }
  if (simulation_data_from_file)
     fscanf(in,"%s",choice);
  else strcpy(choice,scan_input("C",12,45,1,error,&point));
  if (parmfile) fprintf(par,"%s\n",choice);
  if (strcmp(choice,"y") == 0 || strcmp(choice,"Y") == 0)
     put_aphids(in);
  else if (print_data) fprintf(p1,"\n");
 }


change_constants(in)
FILE *in;
/*****************************************************************/
/* Functie:     Inlezen van nieuw gedefineerde waarden voor de   */
/*              constanten van dit programma.                    */
/*****************************************************************/
 {float answer;
  short point, error = 0;
  char choice[2];
  char result[MAX_CHAR];
  /*                                                              */
  /* Inlezen van de opstartwaarden voor de constanten van file.   */
  if (simulation_data_from_file)
   {fscanf(in,"%d",&MAX_AGE);
    fscanf(in,"%d",&RETENTION_TIME);
    fscanf(in,"%d",&ACQUISITION_FEEDING_TIME);
    fscanf(in,"%d",&TRANSMISSION_FEEDING_TIME);
    fscanf(in,"%d",&MIN_FLYING_TEMP);
    fscanf(in,"%d",&MAX_FLYING_TEMP);
    fscanf(in,"%d",&MIN_MOBILITY_TEMP);
    fscanf(in,"%d",&OPTIMUM_MOBILITY_TEMP);
    fscanf(in,"%d",&MAX_MOBILITY_TEMP);
    fscanf(in,"%d",&MIN_LEAF_APPEARANCE_TEMP);
    fscanf(in,"%d",&MIN_LEAF_EXPANSION_TEMP);
```

```
 fscanf(in,"%d",&OPTIMUM_GROWTH_TEMP);
 fscanf(in,"%d",&MAX_GROWTH_TEMP);
 fscanf(in,"%d",&OPTIMUM_MOBILITY_AGE);
 fscanf(in,"%d",&MAX_VELOCITY);
}else
{LOOP{/* screen0 */
 screen0();
 LOOP{
 strcpy(result,scan_input("I",3,45,MAX_CHAR,error,&point));
 if (strlen(result))
  {answer = value(result,point);
   MAX_AGE = answer;
   if (parmfile) fprintf(par,"%d\n",MAX_AGE);
   break;
  }else
 if (MAX_AGE == 0)
  {novalue(); error = 1;
   continue;
  }else break;
 } /* end loop */
 LOOP{
 strcpy(result,scan_input("I",4,44,MAX_CHAR,error,&point));
 if (strlen(result))
  {answer = value(result,point);
   RETENTION_TIME = answer;
   if (parmfile) fprintf(par,"%d\n",RETENTION_TIME);
   break;
  }else
 if (RETENTION_TIME == 0)
  {novalue(); error = 1;
   continue;
  }else break;
 } /* end loop */
 LOOP{
 strcpy(result,scan_input("I",5,47,MAX_CHAR,error,&point));
 if (strlen(result))
  {answer = value(result,point);
   ACQUISITION_FEEDING_TIME = answer;
   if (parmfile) fprintf(par,"%d\n",ACQUISITION_FEEDING_TIME);
   break;
  }else
 if (ACQUISITION_FEEDING_TIME == 0)
  {novalue(); error = 1;
   continue;
  }else break;
 } /* end loop */
 LOOP{
 strcpy(result,scan_input("I",6,47,MAX_CHAR,error,&point));
 if (strlen(result))
  {answer = value(result,point);
   TRANSMISSION_FEEDING_TIME = answer;
   if (parmfile) fprintf(par,"%d\n",TRANSMISSION_FEEDING_TIME);
   break;
  }else
 if (TRANSMISSION_FEEDING_TIME == 0)
  {novalue(); error = 1;
   continue;
  }else break;
```

```
} /* end loop */
LOOP{
strcpy(result,scan_input("I",7,42,MAX_CHAR,error,&point));
if (strlen(result))
 {answer = value(result,point);
  MIN_FLYING_TEMP = answer;
  if (parmfile) fprintf(par,"%d\n",MIN_FLYING_TEMP);
  break;
 }else
if (MIN_FLYING_TEMP == 0)
 {novalue(); error = 1;
  continue;
 }else break;
} /* end loop */
LOOP{
strcpy(result,scan_input("I",8,42,MAX_CHAR,error,&point));
if (strlen(result))
 {answer = value(result,point);
  MAX_FLYING_TEMP = answer;
  if (parmfile) fprintf(par,"%d\n",MAX_FLYING_TEMP);
  break;
 }else
if (MAX_FLYING_TEMP == 0)
 {novalue(); error = 1;
  continue;
 }else break;
} /* end loop */
LOOP{
strcpy(result,scan_input("I",9,41,MAX_CHAR,error,&point));
if (strlen(result))
 {answer = value(result,point);
  MIN_MOBILITY_TEMP = answer;
  if (parmfile) fprintf(par,"%d\n",MIN_MOBILITY_TEMP);
  break;
 }else
if (MIN_MOBILITY_TEMP == 0)
 {novalue(); error = 1;
  continue;
 }else break;
} /* end loop */
LOOP{
strcpy(result,scan_input("I",10,41,MAX_CHAR,error,&point));
if (strlen(result))
 {answer = value(result,point);
  OPTIMUM_MOBILITY_TEMP = answer;
  if (parmfile) fprintf(par,"%d\n",OPTIMUM_MOBILITY_TEMP);
  break;
 }else
if (OPTIMUM_MOBILITY_TEMP == 0)
 {novalue(); error = 1;
  continue;
 }else break;
} /* end loop */
LOOP{
strcpy(result,scan_input("I",11,41,MAX_CHAR,error,&point));
if (strlen(result))
 {answer = value(result,point);
  MAX_MOBILITY_TEMP = answer;
```

```
   if (parmfile) fprintf(par,"%d\n",MAX_MOBILITY_TEMP);
   break;
  }else
 if (MAX_MOBILITY_TEMP == 0)
  {novalue(); error = 1;
   continue;
  }else break;
 } /* end loop */
 LOOP{
 strcpy(result,scan_input("I",12,59,MAX_CHAR,error,&point));
 if (strlen(result))
  {answer = value(result,point);
   MIN_LEAF_APPEARANCE_TEMP = answer;
   if (parmfile) fprintf(par,"%d\n",MIN_LEAF_APPEARANCE_TEMP);
   break;
  }else
 if (MIN_LEAF_APPEARANCE_TEMP == 0)
  {novalue(); error = 1;
   continue;
  }else break;
 } /* end loop */
 LOOP{
 strcpy(result,scan_input("I",13,52,MAX_CHAR,error,&point));
 if (strlen(result))
  {answer = value(result,point);
   MIN_LEAF_EXPANSION_TEMP = answer;
   if (parmfile) fprintf(par,"%d\n",MIN_LEAF_EXPANSION_TEMP);
   break;
  }else
 if (MIN_LEAF_EXPANSION_TEMP == 0)
  {novalue(); error = 1;
   continue;
  }else break;
 } /* end loop */
 LOOP{
 strcpy(result,scan_input("I",14,52,MAX_CHAR,error,&point));
 if (strlen(result))
  {answer = value(result,point);
   OPTIMUM_GROWTH_TEMP = answer;
   if (parmfile) fprintf(par,"%d\n",OPTIMUM_GROWTH_TEMP);
   break;
  }else
 if (OPTIMUM_GROWTH_TEMP == 0)
  {novalue(); error = 1;
   continue;
  }else break;
 } /* end loop */
 LOOP{
 strcpy(result,scan_input("I",15,52,MAX_CHAR,error,&point));
 if (strlen(result))
  {answer = value(result,point);
   MAX_GROWTH_TEMP = answer;
   if (parmfile) fprintf(par,"%d\n",MAX_GROWTH_TEMP);
   break;
  }else
 if (MAX_GROWTH_TEMP == 0)
  {novalue(); error = 1;
   continue;
```

```
      }else break;
     } /* end loop */
     LOOP{
     strcpy(result,scan_input("I",16,52,MAX_CHAR,error,&point));
     if (strlen(result))
      {answer = value(result,point);
       OPTIMUM_MOBILITY_AGE = answer;
       if (parmfile) fprintf(par,"%d\n",OPTIMUM_MOBILITY_AGE);
       break;
      }else
     if (OPTIMUM_MOBILITY_AGE == 0)
      {novalue(); error = 1;
       continue;
      }else break;
     } /* end loop */
     LOOP{
     strcpy(result,scan_input("I",17,49,MAX_CHAR,error,&point));
     if (strlen(result))
      {answer = value(result,point);
       MAX_VELOCITY = answer;
       if (parmfile) fprintf(par,"%d\n",MAX_VELOCITY);
       break;
      }else
     if (MAX_VELOCITY == 0)
      {novalue(); error = 1;
       continue;
      }else break;
     } /* end loop */
     if (terminate()) break;
     else continue;
     }/* end screen0 */
    }
  }


novalue()
 {cursor(20,8);
  printf("Er moet een waarde of een naam opgegeven worden!");
 }


short terminate()
 {char choice[2];
  short point;
  if (!simulation_data_from_file)
   {sascr(BOLD);
    cursor(2,5); printf(EMPTY_STRING);
    cursor(3,5); printf("Moet er nog wat gewijzigd?                ");
    sascr(OFF);
    strcpy(choice,scan_input("C",3,33,1,0,&point));
    if (strcmp(choice,"y") == 0)
     {cursor(3,5); printf(EMPTY_STRING);
      if (parmfile)
       {clrscr();
        cursor(14,8); printf("Bij het maken van een parmfile mogen
        geen fouten gemaakt worden!");
        exit(1);
       }
```

```
      return 0;
     }
    else return 1;
   }else return 1;
 }


init_var()
/*****************************************************************/
/* Functie: Initialiseren van alle globale variabelen die nog niet  */
/*          geinitialiseerd zijn in de functie init_dialog() en     */
/*          die wel geinitialiseerd moeten worden.                  */
/*****************************************************************/
 {int i;
  int dagen = 0;
  char ch;
  FILE *q;
  /*                                                                */
  /* Default setting voor predation.                                */
  predation = 100;
  there_were_no_aphids_yet = 1;
  /*                                                                */
  /* Wegschrijven naar file van de rulers van de grafieken voor     */
  /* luizenaantallen en vergeelde planten.                          */
  if (print_luizengrafiek)
   {fprintf(p3,"%100s\n\n","LUIZENAANTALLEN (*)");
    fprintf(p3,"Datum      0¦");
    for (i = 100; i <= 1000; i += 100)
    fprintf(p3,"%10d",i);
    fprintf(p3,"\n");
    fprintf(p3,"-----------¦");
    for (i = 100; i <= 1000; i += 100)
    fprintf(p3,"----+----+");
    fprintf(p3,"\n");
   }
  if (print_plantengrafiek)
   {fprintf(p4,"%100s\n\n","VERGEELDE PLANTEN (*)");
    fprintf(p4,"Datum      0¦");
    for (i = 100; i <= 1000; i += 100)
    fprintf(p4,"%10d",i);
    fprintf(p4,"\n");
    fprintf(p4,"-----------¦");
    for (i = 100; i <= 1000; i += 100)
    fprintf(p4,"----+----+");
    fprintf(p4,"\n");
   }
  /*                                                                */
  /* Initialiseren van de groeiloze periode.                        */
  growthless_period = 20;
  /*                                                                */
  /* Initialiseren van de dummy aphid.                              */
  dummy_aphid = (struct aphid *) calloc(1,sizeof(struct aphid));
  /*                                                                */
  /* Zaadje gebruiken:                                              */
  for (i = 0; i < seed; i ++)
  rand();
  /*                                                                */
  /* Initialiseren van de temperaturen tabel voor de maanden april  */
```

```
/* t/m oktober.                                                    */
q = openen("temp.dat","r");
for (i = 1; i < TEMP_GEG; i ++)
fscanf(q,"%2d ",(temp + i));
fclose(q);
/*                                                                 */
/* Inlezen van de incubatie tijden.                                */
if (sowing_month < 5)
 {q = openen("incubold.dat","r");
  for (i = 0; i <= 30; i ++)
  fscanf(q,"%4d ",(incubation_time + i));
  fclose(q);
 }else
 {q = openen("incyoung.dat","r");
  for (i = 0; i <= 30; i ++)
  fscanf(q,"%4d ",(incubation_time + i));
  fclose(q);
 }
/*                                                                 */
/* Initialiseren van de groeifactoren per blad.                    */
growth_factor[1] = .1; growth_factor[2] = .1; growth_factor[3] = .3;
growth_factor[4] = .5; growth_factor[5] = .7; growth_factor[6] = .9;
growth_factor[7] = .9; growth_factor[8] = .95; growth_factor[9] = 1;
growth_factor[10] = 1;
for (i = 11; i < 51; i ++)
growth_factor[i] = i/-195.0 + 1.05641;
for (i = 51; i < 80; i ++)
growth_factor[i] = 0.8;
/*                                                                 */
/* Bepalen van het aantal dagen dat ligt tussen 1 april (begin     */
/* v/d temperaturen tabel) en het moment van zaaien. Aan de hand   */
/* hiervan wordt period berekend. Period wijst naar een plaats in  */
/* de temperaturen tabel behorende bij het tijdstip van simulatie.*/
for (i = 4; i < sowing_month; i ++)
dagen += monthdays[i];
dagen += sowing_day - 1;
period = dagen * 8;
dagen = 0;
period_on_sowing_date = period;
/*                                                                 */
/* Bepalen van het aantal dagen dat ligt tussen het moment van     */
/* zaaien en het moment dat de simulatie aan moet vangen. Aan de   */
/* hand van dit aantal dagen wordt period berekend. Ook de lengte  */
/* van de groeiloze periode wordt bijgesteld.                      */
for (i = sowing_month; i < month; i ++)
dagen += monthdays[i];
dagen += day - sowing_day;
if (dagen > 20)
 {growthless_period = 0;
  period += 20 * 8;
  dagen -= 20;
 }else
 {growthless_period -= dagen;
  period += dagen * 8;
  dagen = 0;
 }
/*                                                                 */
/* Aan de hand van het aantal berekende dagen laten we de planten  */
```

```
   /* groeien tot de start datum van de simulatie.                    */
   for (i = 0; i < (dagen * 24); i ++)
    {time = i;
     weather();
     if (i%24 == 0) plant_growth();
    }time = 0;
   period_on_starting_date = period;
  }
```

File simulate.c

```
#include <stdio.h>
#include <ctype.h>
#include <math.h>
#include <malloc.h>
#include <string.h>
#include <stdlib.h>
#include <color.h>

#define POWER(x) (x)*(x)
#define CONSTANT int
#define MAXLENGTH 100
#define MAXWIDTH 50
#define TEMP_GEG 1713
#define MAX_LEAVES 80
#define MAX_APHIDS 1000
#define MAX_CHAR 8
#define EMPTY_STRING "    (70 spaties)    "
#define LOOP while(1)


CONSTANT MAX_AGE;
CONSTANT RETENTION_TIME;
CONSTANT ACQUISITION_FEEDING_TIME;
CONSTANT TRANSMISSION_FEEDING_TIME;
CONSTANT MIN_FLYING_TEMP;
CONSTANT MAX_FLYING_TEMP;
CONSTANT MIN_MOBILITY_TEMP;
CONSTANT OPTIMUM_MOBILITY_TEMP;
CONSTANT MAX_MOBILITY_TEMP;
CONSTANT MIN_LEAF_APPEARANCE_TEMP;
CONSTANT MIN_LEAF_EXPANSION_TEMP;
CONSTANT OPTIMUM_GROWTH_TEMP;
CONSTANT MAX_GROWTH_TEMP;
CONSTANT OPTIMUM_MOBILITY_AGE;
CONSTANT MAX_VELOCITY;


struct aphid {
  int time_of_birth;
  int time_of_contamination;
  int time_of_switch;
  int x, y;
  short winged;
```

```
  struct aphid *next, *pred;
} *first_aphid, *last_aphid;

struct beet {
  int class;
  int time_of_contamination;
  int nbr_of_aphids;
  int healthy_leaves;
  short yellowed;
} *field[MAXLENGTH][MAXWIDTH];


int fieldlength, fieldwidth;
float widthplants, lengthplants;
int widthfraction, lengthfraction;
int tot_widthplants, tot_lengthplants;
int tot_plants, tot_aphids;
int time, maxtime;
int month, day;
int sowing_month, sowing_day;
int monthdays[13];
int temperature, temp[TEMP_GEG];
int print_interval_time;
int field_counting[215];
int incubation_time[31];
int period_on_sowing_date;
int period_on_starting_date;
int growthless_period;
int initial_age;
int aphid_days;
int predation;
int interval;
int period;

float total_cdays;
float mean_temp;
float nbr_of_leaves;
float total_leaf_area;
float growth_factor[MAX_LEAVES];
float final_leaf_area[MAX_LEAVES];
float stage_of_development[MAX_LEAVES];

short parmfile;
short print_data;
short print_picture;
short print_luizengrafiek;
short print_plantengrafiek;
short simulation_data_from_file;
short aphid_data_from_file;
short there_were_no_aphids_yet;
char  *monthname[13];
char  parmfilename[16];
int   seed;

struct aphid *this_aphid;
struct aphid *dummy_aphid;

FILE *fopen();
```

```
FILE *p1, *p2, *p3, *p4, *par;


FILE     *openen(char *,char *);
int      uniform(int);
char     *scan_input(char *,short,short,short,short,short *);
float    value(char *,short);
void     screen(void);
void     screen0(void);
void     screen1(void);
void     screen2(void);
void     screen3(void);
void     screen4(void);
void     screen5(void);
void     screen6(void);
void     screen7(void);
void     screen8(void);
void     create_aphid(int,short,int,int,short);
void     delete_aphid(void);
void     init_dialog(char *);
void     contaminate_plant(FILE *);
void     put_aphids(FILE *);
void     change_constants(FILE *);
void     novalue(void);
short    terminate(void);
void     init_var(void);
void     cast_skin(void);
void     virus_exchange(void);
int      latent_period(void);
short    contaminated_leaf(void);
short    aphid_contaminated(void);
void     symptoms_development(void);
void     distance_covered(void);
void     reproduction(void);
void     aphid_removal(void);
void     plant_growth(void);
void     weather(void);
void     immigration(void);
void     print_image(void);
void     aphid_amount_checking(void);
void     aphid_amount_adjusting(int,int);
void     estimate_predation(void);
void     date(short);
void     rerun(void);
void     main(int,char **);


cast_skin()
/*****************************************************************/
/* Functie: Checken of de luis vervelt. Zo ja, zorgen dat de luis   */
/*          niet meer infectueus is.                               */
/*****************************************************************/
 {int age;
  age = time - this_aphid->time_of_birth;
  if (age < 145 && (age == 36 || age == 72 || age == 98 ||
      age == 144))
  this_aphid->time_of_contamination -= RETENTION_TIME;
 }
```

```
virus_exchange()
/*****************************************************************/
/* Functie: Bepalen of een besmette luis een niet besmette plant    */
/*          infecteerd of dat een luis het virus uit een besmette   */
/*          plant opneemt.                                          */
/*****************************************************************/
 {int i, x, y;
  x = this_aphid->x/lengthfraction;
  y = this_aphid->y/widthfraction;
  if (field[x][y]->time_of_contamination >= 0)
   {if (time - field[x][y]->time_of_contamination > latent_period() &&
        contaminated_leaf() &&
        time - this_aphid->time_of_switch > ACQUISITION_FEEDING_TIME)
       this_aphid->time_of_contamination = time;
    else return;
   }else
  if (aphid_contaminated() &&
      time - this_aphid->time_of_switch > TRANSMISSION_FEEDING_TIME)
   {field[x][y]->time_of_contamination = time;
    field[x][y]->healthy_leaves = floor(nbr_of_leaves);
   }
 }


int latent_period()
/*****************************************************************/
/* Functie: Berekenen van de lengte van de latente periode. Deze    */
/*          waarde wordt als functiewaarde opgeleverd.              */
/*****************************************************************/
 {int periode;
  int i, uren, dagen = 0;
  for (i = sowing_month; i < month; i ++)
  dagen += monthdays[i];
  dagen += day - sowing_day;
  uren = dagen*24 + time;
  periode = 192.0*(uren/5136.0) + 96.0;
  return periode;
 }


short contaminated_leaf()
/*****************************************************************/
/* Functie: Bepalen of een luis zich al of niet op een besmet blad  */
/*          bevindt. Als functiewaarden worden dan respectievelijk  */
/*          een 1 of een 0 opgeleverd.                              */
/*****************************************************************/
 {int i, x, y;
  static short fraction = 1;
  int klasse;
  float besmet_opp = 0.0;
  x = this_aphid->x/lengthfraction;
  y = this_aphid->y/widthfraction;
  /*                                                               */
  /* Berekenen van de klasse waarin de plant zich bevindt.         */
  for (i = floor(nbr_of_leaves);
       i > field[x][y]->healthy_leaves;
```

```
        i --)
   besmet_opp += final_leaf_area[i];
   if (total_leaf_area != 0.0)
   klasse = ceil(besmet_opp/total_leaf_area * 10);
   fraction += klasse;
   if (fraction > 2)
    {fraction -= 2;
     return 1;
    }else
   return 0;
  }


short aphid_contaminated()
/*****************************************************************/
/* Functie: Onderzoeken of de luis vervelt is! Zo ja: return 0;     */
/*          Zo nee: return 1;                                      */
/*****************************************************************/
 {if (time - this_aphid->time_of_contamination > RETENTION_TIME)
     return 0;
  else return 1;
 }


symptoms_development()
/*****************************************************************/
/* Functie: Bepalen van de vergeling van de bietenplanten.         */
/*****************************************************************/
 {int i, j;
  int index;
  for (i = 0; i < tot_lengthplants; i ++)
  for (j = 0; j < tot_widthplants; j ++)
   {if (field[i][j]->yellowed == 0 &&
        field[i][j]->time_of_contamination >= 0)
     {index = period/56.0 - (time - field[i][j]-
>time_of_contamination)/168.0;
      if (time-field[i][j]->time_of_contamination >
incubation_time[index]*24)
      field[i][j]->yellowed = 1;
     }
   }
 }


distance_covered()
/*****************************************************************/
/* Functie: Berekenen van de afgelegde afstand. Bepalen van de hoek */
/*          waaronder de luis loopt. Aan de hand van deze gegevens  */
/*          de nieuwe coordinaten van de luis berekenen.           */
/*****************************************************************/
 {int angle;
  int age, temp;
  float pi_angle;
  int x_bef, y_bef, x_aft, y_aft;
  float divide_velocity, divide_factor;
  float velocity, factor;
  float percentage;
  float total_distance;
```

```
float x_moved, y_moved;
static int fraction = 0;
static float move = 0.0;
x_bef = this_aphid->x/lengthfraction;
y_bef = this_aphid->y/widthfraction;
age = time - this_aphid->time_of_birth;
temp = temperature;
if (temperature < MIN_MOBILITY_TEMP ||
    temperature > MAX_MOBILITY_TEMP) temp = MIN_MOBILITY_TEMP;
/*                                                              */
/* total_distance uitrekenen.                                   */
/* Alleen indien het gewas gesloten is voor alle luizen!        */
/* Zo niet, alleen voor elke 100-e luis.                        */
if (nbr_of_leaves < 8.0)
   percentage = 7.75 * nbr_of_leaves;
else percentage = -0.41 * nbr_of_leaves + 65;
percentage /= 100.0;
/* percentage = 0.6/1700.0 * (period-period_on_sowing_date)+0.2;  */
if (nbr_of_leaves < 12.0 && fraction < 40)
 {total_distance = 0.0;
  fraction ++;
 }else
 {divide_velocity = 360000/(float)MAX_VELOCITY;
  divide_factor = (float)POWER(MAX_MOBILITY_TEMP -
  OPTIMUM_MOBILITY_TEMP);
  velocity = -pow((double)(abs(age -
  OPTIMUM_MOBILITY_AGE)),2.0)/divide_velocity + MAX_VELOCITY;
  factor = -pow((double)(abs(temp -
  OPTIMUM_MOBILITY_TEMP)),2.0)/divide_factor + 1;
  total_distance = velocity * factor * percentage;
  if (fraction >= 40) fraction -= 40;
 }
/*                                                              */
/* angle trekken.                                               */
angle = uniform(360);
pi_angle = angle/180.0 * 3.141592653589;
x_moved = sin(pi_angle) * total_distance;
y_moved = cos(pi_angle) * total_distance;
if (x_moved < 0) x_moved = ceil(x_moved);
else x_moved = floor(x_moved);
if (y_moved < 0) y_moved = ceil(y_moved);
else y_moved = floor(y_moved);
x_aft = (this_aphid->x + x_moved)/lengthfraction;
y_aft = (this_aphid->y + y_moved)/widthfraction;
/* printf("this_aphid->x: %d\n",this_aphid->x);
   printf("this_aphid->y: %d\n",this_aphid->y);
   printf("x_bef: %d\n",x_bef);
   printf("y_bef: %d\n",y_bef);
   cursor(23,1);
   printf("div vel, div fac, vel, fac: %f %f %f %f",
   divide_velocity,divide_factor,velocity,factor);
   cursor(24,1);
   printf("total_distance age temp percentage: %f %d %d %f",
   total_distance,age,temp,percentage);
   printf("angle: %d\n",angle);
   printf("pi_angle: %f\n",pi_angle);
   printf("x_moved: %f\n",x_moved);
   printf("y_moved: %f\n",y_moved);
```

```
      printf("x_aft: %d\n",x_aft);
      printf("y_aft: %d\n\n",y_aft);                                    */
   /*                                                                   */
   /* Opdat de luis niet buiten de coordinaten van het veld zal lopen*/
   /* worden hier de grenzen bewaakt. Loopt een luis tegen de grens  */
   /* op dan wordt ze gestopt.                                          */
   if (x_aft >= tot_lengthplants)
    {x_aft = tot_lengthplants - 1;
     this_aphid->x = tot_lengthplants * lengthfraction - 1;
    }else
   if (this_aphid->x + x_moved < 0)
    {x_aft = 0;
     this_aphid->x = 0;
    }else
   this_aphid->x += x_moved;
   if (y_aft >= tot_widthplants)
    {y_aft = tot_widthplants - 1;
     this_aphid->y = tot_widthplants * widthfraction - 1;
    }else
   if (this_aphid->y + y_moved < 0)
    {y_aft = 0;
     this_aphid->y = 0;
    }else
   this_aphid->y += y_moved;
   /*                                                                   */
   /* Indien de luis van plant wisselt, moet het aantal luizen van de*/
   /* betreffende planten aangepast worden.                             */
   if (x_bef != x_aft || y_bef != y_aft)
    {field[x_bef][y_bef]->nbr_of_aphids --;
     field[x_aft][y_aft]->nbr_of_aphids ++;
    }
   /*                                                                   */
   /* Time_of_switch moet gelijk gemaakt worden aan time indien er   */
   /* loopactiviteit is geweest. In oud gewas is de loopactiviteit   */
   /* hoger dan in jong gewas.                                          */
   move += percentage;
   if (move > 1.0)
    {move -= 1.0;
     this_aphid->time_of_switch = time;
    }
 }


reproduction()
/*****************************************************************/
/* functie opgenomen voor het geval dat in een later stadium beslo- */
/* ten wordt om een luizenpopulatiemodel in dit model te verwerken. */
/*****************************************************************/
 {}


aphid_removal()
/*****************************************************************/
/* Functie: Verwijderen van te oude luizen, gepredeerde luizen en   */
/*          weggevlogen luizen uit de simulatie.                     */
/*****************************************************************/
 {int x, y;
  static short fraction = 0;
```

```
  short winged;
  fraction ++;
  x = this_aphid->x;
  y = this_aphid->y;
  /* estimate_predation();                                          */
  winged = this_aphid->winged;
  if (winged &&
      temperature > MIN_FLYING_TEMP &&
      temperature < MAX_FLYING_TEMP &&
      time - this_aphid->time_of_birth > 144)
   {create_aphid(0,0,x,y,winged);
    delete_aphid();
   }else
  if (time - this_aphid->time_of_birth > MAX_AGE*24)
   {create_aphid(0,0,x,y,winged);
    delete_aphid();
   }else
  if (fraction > predation)
   {fraction -= predation;
    create_aphid(0,0,x,y,winged);
    delete_aphid();
   }
 }


plant_growth()
/*****************************************************************/
/* Functie: Berekenen van de groei van de plant aan de hand van de  */
/*          heersende temperatuur.                                   */
/*****************************************************************/
 {int temp, i;
  int leaf_number;
  int cdays_of_development;
  float leaf_expansion, appeared_leaves;
  if (growthless_period > 0) growthless_period --;
  else
   {if (mean_temp > OPTIMUM_GROWTH_TEMP)
       temp = MAX_GROWTH_TEMP - mean_temp;
    else temp = mean_temp;
    if (mean_temp > MAX_GROWTH_TEMP ||
        mean_temp < MIN_LEAF_APPEARANCE_TEMP)
       temp = MIN_LEAF_APPEARANCE_TEMP;
    /*                                                              */
    /* Bepalen van het aantal verschenen bladeren.                  */
    appeared_leaves = 3.16e-2 * temp - 3.16e-2;
    nbr_of_leaves += appeared_leaves;
    leaf_number = floor(nbr_of_leaves);
    for (i = 1; i <= leaf_number; i ++)
     {/*                                                            */
      /* Bepalen van het aantal graaddagen van ontwikkeling.       */
      cdays_of_development = (4.25 * temp + 355) * growth_factor[i];
      if (stage_of_development[i] < 1)
       {if (mean_temp < MIN_LEAF_EXPANSION_TEMP ||
            mean_temp > MAX_GROWTH_TEMP)
           temp = MIN_LEAF_EXPANSION_TEMP;
        /*                                                          */
        /* Bepalen van de hoeveelheid blad wat bijgroeit.           */
        leaf_expansion = growth_factor[i] * (1.1 * temp - 3.3);
```

```
          final_leaf_area[i] += leaf_expansion;
          total_leaf_area += leaf_expansion;
          if (cdays_of_development != 0)
          stage_of_development[i] += temp/(float)cdays_of_development;
         }
       }
     }
  }


weather()
/*****************************************************************/
/* Functie: Bepalen van period. Aan de hand van period wordt de  */
/*          juiste temperatuur uit de temperaturen tabel gehaald.*/
/*****************************************************************/
 {int i;
  if (time%3 == 0)
   {period ++;
    temperature = temp[period];
    total_cdays += temperature/8.0;
    if (time%24 == 0)
     {aphid_days += tot_aphids;
      mean_temp = 0.0;
      for (i = 0; i < 8; i ++)
      mean_temp += (float)temp[period + i];
      mean_temp /= 8.0;
     }
   }
 }


immigration()
/*****************************************************************/
/* Voorlopig geen immigratie. We veronderstellen in dit stadium  */
/* een vast of gefixed aantal luizen in het veld.                */
/*****************************************************************/
 {}


print_image()
/*****************************************************************/
/* Functie: Printen naar file van de resultaten van de simulatie.*/
/*****************************************************************/
 {int L1_aphid = 0, L2_aphid = 0, L3_aphid = 0, L4_aphid = 0;
  int adult_aphid_1 = 0, adult_aphid_2 = 0;
  int contaminated_plants = 0;
  int contaminated_aphids = 0;
  int yellowed_plants = 0;
  int winged_aphids = 0;
  int i, j, x, y;
  int dagen, aantal_bladeren;
  static int n_prints = 0;
  char info[40];
  int klasse;
  int age;
  float besmet_opp = 0.0;
  char ch[5];
  /*                                                             */
```

```
/* Printen van het aantal plaatjes dat geprint is.                    */
sprintf(info,"Plaatje %d is geprint.",++ n_prints);
for (i = 0; i <= 21; i ++)
 {cursor(2+i,75);
  printf("%c",info[i]);
 }
/*                                                                    */
/* Aanroepen van symptoms_development() om de vergeling in het        */
/* veld te bepalen.                                                   */
symptoms_development();
/*                                                                    */
/* Lijst van luizen doorlopen om alle bijzonderheden van de luizen*/
/* vast te leggen.                                                    */
for (this_aphid = first_aphid;
     this_aphid != NULL;
     this_aphid = this_aphid->next)
 {age = time - this_aphid->time_of_birth;
  if (time - this_aphid->time_of_contamination < RETENTION_TIME)
     contaminated_aphids ++;
  if (this_aphid->winged) winged_aphids ++;
  if (age < 36) L1_aphid ++;
  else if (age < 72) L2_aphid ++;
  else if (age < 108) L3_aphid ++;
  else if (age < 144) L4_aphid ++;
  else if (age < 360) adult_aphid_1 ++;
  else adult_aphid_2 ++;
 }
/*                                                                    */
/* Berekenen van de klasse waarin de planten zich bevinden.           */
for (x = 0; x < tot_lengthplants; x ++)
for (y = 0; y < tot_widthplants; y ++)
 {if (field[x][y]->yellowed) yellowed_plants ++;
  for (i = floor(nbr_of_leaves);
       i > field[x][y]->healthy_leaves;
       i --)
  besmet_opp += final_leaf_area[i];
  if (besmet_opp != 0.0 && total_leaf_area != 0.0)
   {klasse = ceil(besmet_opp/total_leaf_area * 10);
    field[x][y]->class = klasse;
    contaminated_plants ++;
    besmet_opp = 0.0;
   }
 }
date(0);
/*                                                                    */
/* Algemene informatie over de simulatie.                             */
if (print_data)
 {/* fprintf(p1,"\t\tAantal vergeelde planten: %d.\n",
  yellowed_plants);*/
  fprintf(p1,"\nTemperatuur: %d graden.",temperature);
  fprintf(p1,"\t\t\tTotale aantal bladluizen: %d.\n",tot_aphids);
  fprintf(p1,"Aantal besmette bladluizen:   %d.",
  contaminated_aphids);
  fprintf(p1,"\tAantal gevleugelde bladluizen: %d.\n",
  winged_aphids);
  /* fprintf(p1,"Totale aantal bietenplanten: %d.",tot_plants);   */
  /* fprintf(p1,"\tSimulatieduur: %d dagen.\n",time/24);          */
  aantal_bladeren = floor(nbr_of_leaves);
```

```
  fprintf(p1,"Aantal bladeren per plant: %d.",aantal_bladeren);
  fprintf(p1,"\t\tTotale bladoppervlak: %f.\n",total_leaf_area);
  /* fprintf(p1,"\n");
     fprintf(p1,"Luizen in L1 stadium: %d  ",L1_aphid);
     fprintf(p1,"\t\tLuizen in L4 stadium: %d\n",L4_aphid);
     fprintf(p1,"Luizen in L2 stadium: %d  ",L2_aphid);
     fprintf(p1,"\t\tAdulten (tot 360 uur): %d\n",adult_aphid_1);
     fprintf(p1,"Luizen in L3 stadium: %d  ",L3_aphid);
     fprintf(p1,"\t\tAdulten (tot 1008 uur): %d\n",adult_aphid_2);*/
  fprintf(p1,"Luisdagen: %d :: %d %d %d %d %d %d :: Predatie: %d\n",
  aphid_days,L1_aphid,L2_aphid,L3_aphid,L4_aphid,
  adult_aphid_1,adult_aphid_2,predation);
  fprintf(p1,"\n");
 }
/*                                                                  */
/* Printen van de verschillende klassen van de planten.             */
if (print_picture)
 {for (i = 0; i < tot_lengthplants+2; i ++)
  fprintf(p2,"-"); fprintf(p2,"\n");
  for (j = tot_widthplants - 1; j >= 0; j --)
   {fprintf(p2,"¦");
    for (i = 0; i < tot_lengthplants; i ++)
     {if (field[i][j]->yellowed)
       {sascr(BOLD);
        if (field[i][j]->class > 9) fputc(219,p2);
        else if (field[i][j]->class > 6) fputc(178,p2);
        else if (field[i][j]->class > 3) fputc(177,p2);
        else if (field[i][j]->class > 0) fputc(176,p2);
        else fputc(' ',p2);
        sascr(OFF);
       }else
       {if (field[i][j]->class > 9) fputc('T',p2);
        else if (field[i][j]->class > 0)
        fprintf(p2,"%d",field[i][j]->class);
        else fputc(' ',p2);
       }
     }
    fprintf(p2,"¦\n");
   }
  for (i = 0; i < tot_lengthplants+2; i ++)
  fprintf(p2,"-"); fprintf(p2,"\n");
  /*                                                                */
  /* Printen van de verschillende luizenaantallen in het veld per */
  /* plant.                                                         */
  for (i = 0; i < tot_lengthplants+2; i ++)
  fprintf(p2,"-"); fprintf(p2,"\n");
  for (j = tot_widthplants - 1; j >= 0; j --)
   {fprintf(p2,"¦");
    for (i = 0; i < tot_lengthplants; i ++)
     {if (field[i][j]->nbr_of_aphids > 9) fputc(176,p2);
      else if (field[i][j]->nbr_of_aphids > 0)
      fprintf(p2,"%d",field[i][j]->nbr_of_aphids);
      else fprintf(p2," ");
     }fprintf(p2,"¦\n");
   }
  for (i = 0; i < tot_lengthplants+2; i ++)
  fprintf(p2,"-"); fprintf(p2,"\n");
 }
```

```
  /*                                                                  */
  /* Printen van de verschillende klassen planten naar terminal.      */
  cursor(0,0);
  for (j = tot_widthplants - 1; j >= 0; j --)
   {printf("¦");
    for (i = 0; i < tot_lengthplants; i ++)
     {if (field[i][j]->yellowed)
       {sascr(BOLD);
        if (field[i][j]->class > 9) putchar(219);
        else if (field[i][j]->class > 6) putchar(178);
        else if (field[i][j]->class > 3) putchar(177);
        else if (field[i][j]->class > 0) putchar(176);
        else printf(" ");
        sascr(OFF);
       }else
       {if (field[i][j]->class > 9) putchar('T');
        else if (field[i][j]->class > 0)
        printf("%d",field[i][j]->class);
        else printf(" ");
       }
     }
    printf("¦\n");
   }
  /*                                                                  */
  /* Printen van de grafiek van vergeelde planten in het veld.        */
  if (print_plantengrafiek && time != 0)
  if (yellowed_plants == 0)
     fprintf(p4,"*(%d) C%d\n",yellowed_plants,contaminated_plants);
  else
   {fprintf(p4,"¦");
    for (i = 1; i < yellowed_plants/10; i ++)
    fprintf(p4," ");
    fprintf(p4,"*(%d) C%d\n",yellowed_plants,contaminated_plants);
   }
 }


aphid_amount_checking()
/*******************************************************************/
/* Functie: Tellen van het aantal luizen op de middelste 39 planten.*/
/*          Indien dit aantal verschilt van het aantal dat in het  */
/*          veld geteld is, op dezelfde 39 planten, moet het aantal */
/*          luizen in het veld bijgesteld worden.                  */
/*          Dit laatste gebeurt in aphid_amount_adjusting().       */
/*******************************************************************/
 {int i, j;
  int index, deviation;
  int vivo_aphids, simulation_aphids = 0;
  index = period/8;
  /*                                                                */
  /* Bepalen van het aantal luizen dat in het veld op de middelste  */
  /* 39planten geteld is.                                           */
  vivo_aphids = field_counting[index];
  deviation = (vivo_aphids * 10)/100;
  /*                                                                */
  /* Bepalen van het aantal luizen dat in de simulatie op de        */
  /* middelste 39 planten zit.                                      */
  for (i = 22; i <= 26; i ++)
```

```
  for (j = 10; j <= 14; j ++)
  simulation_aphids += field[i][j]->nbr_of_aphids;
  for (i = 23; i <= 25; i ++)
   {simulation_aphids += field[i][9]->nbr_of_aphids;
    simulation_aphids += field[i][15]->nbr_of_aphids;
   }
  for (j = 11; j <= 13; j ++)
   {simulation_aphids += field[21][j]->nbr_of_aphids;
    simulation_aphids += field[27][j]->nbr_of_aphids;
   }
  simulation_aphids += field[20][12]->nbr_of_aphids;
  simulation_aphids += field[28][12]->nbr_of_aphids;
  /*                                                              */
  /* Printen datum en resultaten tellingen.                       */
  if (print_luizengrafiek || print_plantengrafiek)
     date(1);
  if (print_luizengrafiek)
   {if (tot_aphids == 0)
       fprintf(p3,"*(%d) S%d V%d\n",
       tot_aphids,simulation_aphids,vivo_aphids);
    else
     {fprintf(p3,"|");
      for (i = 1; i < tot_aphids/10; i ++)
      fprintf(p3," ");
      fprintf(p3,"*(%d) S%d V%d\n",
      tot_aphids,simulation_aphids,vivo_aphids);
     }
   }
  if (print_plantengrafiek && time%168 != 0) fprintf(p4,"|\n");
  /*                                                              */
  /* Bepalen of het verschil tussen het aantal luizen in de simula- */
  /* tie niet meer dan +/- 10% verschilt van het aantal in het veld */
  if (simulation_aphids > vivo_aphids - deviation &&
      simulation_aphids < vivo_aphids + deviation)
     return;
  else aphid_amount_adjusting(simulation_aphids,vivo_aphids);
 }


aphid_amount_adjusting(old,new)
int old, new;
/*****************************************************************/
/* Functie:  Het updaten van de hoeveelheid luizen in het veld aan  */
/*           de hand van de gegevens uit aphid_amount_checking()    */
/*****************************************************************/
 {int i, j, k, n, nbr_of_aphids;
  int x_plant, y_plant;
  int index = period/8;
  float factor, aphids_calculated, fraction = 0.0;
  /*                                                              */
  /* Indien er geen luizen geteld zijn in de simulatie en wel in het*/
  /* veld:                                                        */
  if (old == 0 && new != 0)
   {for (i = 1; i <= new; i ++)
     {x_plant = 23 * lengthfraction + uniform(lengthfraction);
      y_plant = 11 * widthfraction + uniform(widthfraction);
      create_aphid(initial_age,1,x_plant,y_plant,0);
     }
```

```
   if (field_counting[index-1] == 0 && there_were_no_aphids_yet)
   if (new < 30)
   for (i = 0; i < 2; i ++)
    {x_plant = 23 * lengthfraction + uniform(lengthfraction);
     y_plant = 11 * widthfraction + uniform(widthfraction);
     create_aphid(initial_age,1,x_plant,y_plant,0);
    }
   else if (new < 200)
   for (i = 0; i < 9; i ++)
    {x_plant = 23 * lengthfraction + uniform(lengthfraction);
     y_plant = 11 * widthfraction + uniform(widthfraction);
     create_aphid(initial_age,1,x_plant,y_plant,0);
    }
   else for (i = 0; i < 65; i ++)
    {x_plant = 23 * lengthfraction + uniform(lengthfraction);
     y_plant = 11 * widthfraction + uniform(widthfraction);
     create_aphid(initial_age,1,x_plant,y_plant,0);
    }there_were_no_aphids_yet = 0;
 }
/*                                                                    */
/* Indien er geen luizen geteld zijn in het veld:                     */
else if (new == 0)
   for (this_aphid = first_aphid;
        this_aphid != NULL;
        this_aphid = this_aphid->next)
   delete_aphid();
else
/*                                                                    */
/* In elk ander geval:                                                */
 {factor = new/(float)old;
  if (factor * tot_aphids >= 900.0)
   {factor = 900.0/(float)tot_aphids;
    if (print_data)
     {fprintf(p1,"\n!! Warning !!\n");
      fprintf(p1,"Aantal luizen is kunstmatig onder de 900
      gehouden!\n\n");
     }
   }
  for (i = 0; i < tot_lengthplants; i ++)
  for (j = 0; j < tot_widthplants; j ++)
   {nbr_of_aphids = field[i][j]->nbr_of_aphids;
    if (nbr_of_aphids != 0)
     {aphids_calculated = nbr_of_aphids * factor + fraction;
      n = aphids_calculated;
      fraction = aphids_calculated - n;
      while(n > nbr_of_aphids)
       {x_plant = i * lengthfraction + uniform(lengthfraction);
        y_plant = j * widthfraction + uniform(widthfraction);
        create_aphid(0,0,x_plant,y_plant,0);
        n --;
       }
      this_aphid = first_aphid;
      while(n < nbr_of_aphids && this_aphid != NULL)
       {if (this_aphid->x/lengthfraction == i &&
            this_aphid->y/widthfraction == j)
         {delete_aphid();
          n ++;
         }this_aphid = this_aphid->next;
```

```
          }
        }
      }
    }
  }


estimate_predation()
/*****************************************************************/
/* Functie: Schatten hoe hoog de predatie in het veld is geweest.   */
/*          Deze functie is nog in een experimenteel stadium.       */
/*          9-7-87                                                  */
/*****************************************************************/
 {float predatie;
  int x = this_aphid->x/lengthfraction;
  int y = this_aphid->y/widthfraction;
  predatie = field[x][y]->nbr_of_aphids/3.33;
  if (total_leaf_area != 0.0)
  predatie *= 8000/total_leaf_area;
  if (predatie > 4.0) predatie = 4.0;
  if (predatie > 0.0)
  predation = 1/predatie * 100;
  else predation = 1000;
 }


date(grafiek)
short grafiek;
/*****************************************************************/
/* Functie: Bepalen van de datum aan de hand van de startdatum van  */
/*          de simulatie en het aantal uren dat de simulatie loopt. */
/*****************************************************************/
 {int maand, dag, tijd;
  maand = month;
  dag = day;
  tijd = time;
  while (tijd >= (monthdays[maand] - dag+1)*24)
   {tijd -= (monthdays[maand] - dag+1)*24;
    dag = 1;
    maand ++;
   }
  dag += tijd/24;
  tijd %= 24;
  if (!grafiek)
   {if (print_data)
       fprintf(p1,"Datum:   %d %s %d.00 uur.",
       dag,monthname[maand],tijd);
   }else if (time%168 == 0)
   {if (print_luizengrafiek)
       fprintf(p3,"%2d%10s",dag,monthname[maand]);
    if (print_plantengrafiek)
       fprintf(p4,"%2d%10s",dag,monthname[maand]);
   }else
   {if (print_luizengrafiek) fprintf(p3,"             ");
    if (print_plantengrafiek) fprintf(p4,"             ");
   }
 }
```

```
rerun()
/*****************************************************************/
/* Functie: Vrijmaken van geheugenruimte en initialiseren van    */
/*          variabelen om een nieuwe simulatie te kunnen beginnen.*/
/*****************************************************************/
 {short i, x, y;
  for (x = 0; x < tot_lengthplants; x ++)
  for (y = 0; y < tot_widthplants; y ++)
  free(field[x][y]);
  for (this_aphid = first_aphid;
       this_aphid != NULL;
       this_aphid = this_aphid->next)
  delete_aphid();
  time = 0;
  aphid_days = 0;
  total_cdays = 0.0;
  nbr_of_leaves = 0.0;
  total_leaf_area = 0.0;
  for (i = 0; i < MAX_LEAVES; i ++)
   {final_leaf_area[i] = 0.0;
    stage_of_development[i] = 0.0;
   }
 }


main(argc,argv)
int argc;
char **argv;
/*****************************************************************/
/*                                                               */
/*                          Het hoofdprogramma.                  */
/*                                                               */
/*****************************************************************/
 {short i;
  char filename1[16];
  char filename2[16];
  char filename3[16];
  char filename4[16];
  if (argc == 1)
   {simulation_data_from_file = 0;
    argc = 2;
   }else
  simulation_data_from_file = 1;
  for (i = 1; i < argc; i ++)
   {sprintf(filename1,"sim%d.dat",i);
    sprintf(filename2,"sim%d.pic",i);
    sprintf(filename3,"sim%d.lui",i);
    sprintf(filename4,"sim%d.pla",i);
    /*                                                           */
    /* Openen van de files waar alle informatie naar geschreven  */
    /* wordt.                                                    */
    p1 = openen(filename1,"w");
    p2 = openen(filename2,"w");
    p3 = openen(filename3,"w");
    p4 = openen(filename4,"w");
    init_dialog(argv[i]);
    /*                                                           */
```

```
    /* Doorlopen van de simulatie periode.                              */
    if (!parmfile)
    for (time = 0; time <= maxtime; time ++)
     {weather();
      if (time%24 == 0) plant_growth();
      if (time >= interval*24 && time%24 == 0 &&
      aphid_data_from_file) aphid_amount_checking();
      if (time%print_interval_time == 0) print_image();
      /* immigration();                                                 */
      /*                                                                */
      /* Doorlopen van de double linked list van luizen.                */
      for (this_aphid = first_aphid;
           this_aphid != NULL;
           this_aphid = this_aphid->next)
       {/* reproduction();                                             */
        cast_skin();
        virus_exchange();
        distance_covered();
        aphid_removal();
       }
     }
    fclose(p1);
    fclose(p2);
    fclose(p3);
    fclose(p4);
    if (simulation_data_from_file) rerun();
   }
  clrscr();
}
```