Model voor de groei van het cytomegalovirus

Willem Waasdorp

20 juni 2016

Inleiding

Deze scriptie gaat over het Cytomegalovirus. Voortaan zal ik de afkorting CMV gebruiken. Vanuit het UMC kwam de vraag om een model op te stellen om daarmee de resistentie van CMV te bepalen. In deze scriptie zal dat model ontwikkelt worden. Eerst zal ik echter het CMV virus beschrijven, zeggen wat de gevolgen zijn van een infectie met CMV, de mogelijke behandeling weergeven en duidelijk maken waarom het van belang is om een model te ontwikkelen. Daarna zal ik wat vertellen over het experiment in het laboratorium van het UMC, waarvan we de data gebruiken. Vervolgens zal ik het model beschrijven en uitwerken. Met behulp van de data worden dan vervolgens de parameters geschat. Ook zal ik daarbij een stukje theorie vermelden.

Het virus

In dit hoofdstuk zal ik wat meer zeggen over CMV. De informatie komt uit de LCI-richtlijn CMV-infectie van het RIVM.

Het Cytomegalovirus is een virus uit de familie van de herpesvirussen. Hieronder vallen onder andere ook de virussen die verantwoordelijk zijn voor een koortslip, herpes en de ziekte van Pheiffer. Deze virussen blijven na een infectie in het lichaam aanwezig. Ook CMV is een virus wat je lang ongemerkt kan hebben. Het wordt actief op het moment dat het immuunsysteem van de drager verzwakt is. Daarbij gaat het bijvoorbeeld om AIDS-patiënten, personen die een orgaan hebben gekregen en baby's. Bij het grote publiek is weinig bekend over CMV en de gevolgen ervan.

CMV-infecties komen wereldwijd voor, met name in Afrika, Azië en Zuid-Amerika is de verspreiding algemeen. In het Westen wat minder. Maar ook bijvoorbeeld in Nederland ligt het percentage vrouwen op vruchtbare leeftijd die drager zijn van CMV op ongeveer 40%. Bij vrouwen afkomstig uit het Caribisch gebied en het Middellandse Zeegebied ligt dat een stuk hoger. CMV wordt in het algemeen overgedragen via direct en indirect contact van slijmvliezen met besmette lichaamsvloeistoffen zoals speeksel, urine, sperma, bloed en moedermelk. Tijdens de zwangerschap kan het virus ook overgedragen worden van moeder op kind via de placenta. Dit wordt een congenitale besmetting genoemd.

Een besmetting met CMV verloopt meestal ongemerkt. Gezonde mensen zullen hooguit last hebben van koorts of vermoeidheid. Voor mensen met een verzwakt immuunsysteem zoals AIDS-patiënten of mensen die een orgaantransplantatie hebben ondergaan, zijn de symptomen heftiger. Bij deze mensen kan CMV ernstige infecties veroorzaken. Ook bij besmettingen tijdens of vlak na de zwangerschap kunnen

er ernstige gevolgen zijn. Ongeveer 0,6 tot 2,0% van de baby's in Westerse landen wordt geboren met een CMV-infectie. Van de groep baby's die geïnfecteerd geboren wordt heeft ongeveer 17% blijvende schade. Één van de gevolgen is het optreden van doofheid. Bij bijna een kwart van de dove kinderen is een CMV-infectie de oorzaak. De kans op gehoorverlies op de leeftijd van zeven jaar voor kinderen die besmet zijn met CMV in de baarmoeder is ongeveer 10%. Andere problemen zijn bijvoorbeeld een klein hoofdje en problemen met het zicht. Ook kunnen deze kinderen later zowel een geestelijke als lichamelijke ontwikkelingsachterstand vertonen. Van de baby's die bij de geboorte al duidelijke neurologische problemen hebben overlijdt ongeveer 20% in het eerste levensjaar, een goede hygiëne is daarom tijdens de zwangerschap van belang, om zo de kans om zelf besmet te worden te verkleinen.

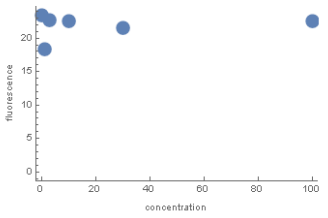
De behandeling van het virus beperkt zich vaak tot de risicogroepen. Bij mensen met een verzwakt immuunsysteem kunnen daarvoor antivirale middelen worden gebruikt. Deze kunnen ook preventief gebruikt worden. Bij pasgeborenen die symptomen van CMV hebben kunnen ook antivirale middelen toegediend worden. Daarmee zou de gehoorschade beperkt kunnen worden. In deze scriptie speelt het antivirale middel gancyclovir een rol. Dit middel remt de voortplanting van de virusdeeltjes in de cellen. Bij de voortplanting van een virus worden de bouwstenen van een virus gerepliceerd. Bouwstenen zijn, o.a., het DNA en de capsule van het virus. Ganciclovir beïnvloedt de DNA-replicatie. Bij het maken van een kopie van het DNA zijn er nucleotiden nodig. Dit zijn bouwstenen van het DNA. Een eiwit leest een streng van het DNA en pakt de juiste nucleotide om de kopie te maken. Ganciclovir remt de voorplanting omdat het door het eiwit kan worden aangezien voor een nucleotide. Op het moment dat het eiwit ganciclovir pakt stopt het kopieër proces e is de DNA-replicatie niet succesvol.

Niet elke patiënt heeft precies het zelfde virus. Het virus kan in verschillende variaties voorkomen. We noemen die variaties stammen van het virus. Niet elke stam reageert hetzelfde op ganciclovir. Het is daarom voor een arts van belang om te weten of de stam die een patiënt heeft goed of slecht reageert op het antivirale middel. Oftewel de mate waarin het eiwit ganciclovir herkent als niet een nucleotide. De vraag die gesteld wordt is: Heeft het zin om het middel toe te dienen? En zo ja, in welke hoeveelheid. De methode die daar tot nu toe voor gebruikt wordt laat wereldwijd een grote variatie zien. Met het ontwikkelen van een nieuwe methode hoopt het UMC betrouwbare resultaten te krijgen.

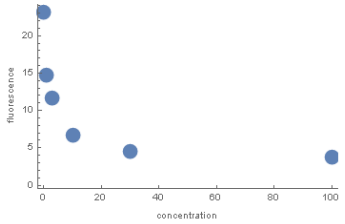
De data

In dit hoofdstuk zal ik de procedure van het UMC weergeven. Deze procedure geeft als uiteindelijk resultaat het percentage cellen waarin het virus zich voortplant ten opzichte van alle cellen. Dit proces zal voor een deel als voorbeeld dienen voor het model.

De eerste stap begint met een kleine hoeveelheid van het virus. Bijvoorbeeld uit een bloedmonster van een patiënt. Deze kleine hoeveelheid wordt gebruikt om het virus te kweken. Het virus wordt samengebracht met gezonde cellen en een voedingsmiddel. Zodra er genoeg van het virus is kan het experiment beginnen. Hierbij worden besmette cellen opnieuw met gezonde cellen gemixt. Dat mengsel wordt in een 6-well plate gestopt. Dat is een plaat met 6 bakjes, met in die bakjes een voedingsmiddel. In het begin is dit een een suspensie maar naar verloop van tijd zakken de cellen naar beneden en vormen een rooster. Het virus kan niet overleven buiten een cel. Dus een besmette cel kan alleen zijn buurcellen besmetten. In het mengsel kan ook een bepaalde concentratie van ganciclovir worden toegediend om te kijken hoe deze stam van het virus er op reageert. Om dat te weten te komen moet de voorplanting van het virus gemeten worden. Daarvoor wordt de FACS-methode gebruikt. FACS betekend fluorescence activated cell sorting, oftewel het sorteren van cellen op basis van fluorescentie van cellen. In dit geval wordt dat gedaan door na drie dagen het mengsel te fixeren, dat betekent dat er geen activiteit meer kan plaatsvinden in het mengsel, en daarna de fluorescentie van een antigeen die aanwezig is als het virus zich voortplant te meten. Voor elke cel kan dan bekeken worden of er voortplanting van het virus aanwezig is. Als de stam van het virus gevoelig is voor ganciclovir en er is voldoende ganciclovir aanwezig dan zal het virus zich minder goed voortplanten en zal er minder fluorescentie zijn. Het uiteindelijke resultaat van het experiment is een percentage van cellen waar voortplanting is bij een bepaalde concentratie van het antivirale middel. In onderstaande afbeelding is een voorbeeld te zien van het resultaat voor zowel een resistente als een gevoelige stam.



Figuur 1: Resistente stam.



Figuur 2: Gevoelige stam.

Het model

Normaal gesproken werd met het resultaat van het hierboven beschreven proces in afwezigheid van ganciclovir het volgende gedaan: Er wordt bepaald bij welke ganciclovir-concentratie virale replicatie met 50% wordt vermindert ten opzichte van de replicatie zonder ganciclovir. Dat komt overeen met een vermindering van 50% in fluorescentie. Deze concentratie wordt de IC50 (inhibitory concentration 50) genoemd.

Één van de problemen daarmee is de enorme variabiliteit in de IC50 tussen verschillende experimenten, wat het vergelijk van IC50s tussen verschillende laboratoria onmogelijk maakt. Met het model wordt geprobeerd om een betere manier te hebben om de resistentie te bepalen.

Het model simuleert de fase van het experiment vanaf het moment dat het mengsel in de bakjes wordt gestopt en gaat over 1 van die bakjes. Aangenomen wordt dat het rooster waar de cellen in liggen een vierkantsrooster is. Zodanig dat je dus de buren kunt bepalen(onder, boven, links en rechts).

We nemen een vierkantsrooster met afmetingen 500 bij 500 met periodieke randvoorwaarden oftewel een torus. Initieel is een fractie x van alle cellen besmet. We veronderstellen dat elke cel onafhankelijk van de andere cellen een kans x heeft om initieel besmet te zijn. In deze initieel besmette cellen is al een hoeveelheid viraal DNA gemaakt en ook de omhulsels van het virus zijn al aangemaakt. Daarom zullen deze cellen hun buren besmetten, zelfs als er een zeer hoge concentratie GCV wordt toegediend. Voor het gemak nemen we aan dat dit direct gebeurt, oftewel alle buren van de initieel besmette buren zijn ook besmet. Dit is de startconfiguratie.

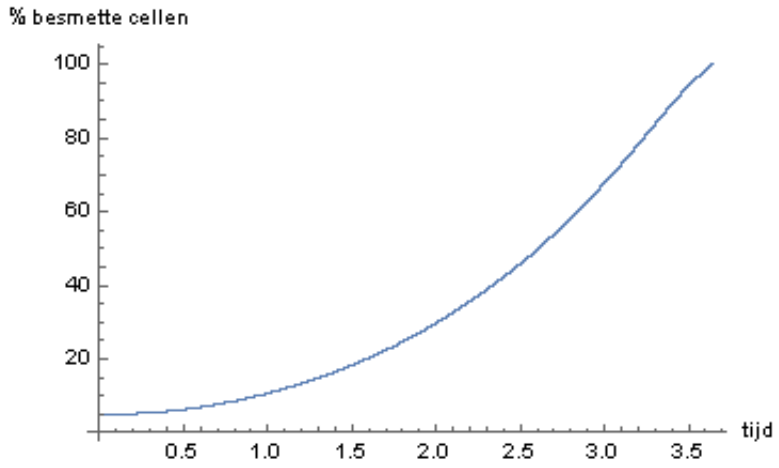
Een besmette cel en een onbesmette buurcel zien we als een paar. Een cel kan nadat hij besmet is niet direct zijn buurcellen besmetten. Het duurt namelijk een tijdje voordat er genoeg virusdeeltjes zijn gemaakt. De tijd tussen de besmetting en het zelf besmettelijk worden noemen we de latente periode. In die periode kan een besmette cel dus nog geen andere cellen besmetten. Zodra de latente periode voorbij is neemt de besmettelijkheid lineair toe. Over de tijd wordt een paar dus besmettelijker. De besmettelijkheid van alle paren bij elkaar opgeteld is de besmettingsdruk in het systeem. Hoe hoger die besmettingsdruk is hoe groter de kans op een nieuwe besmetting, hoe kleiner de tijd tussen nieuwe besmettingen.

Als de startconfiguratie klaar is gaat de tijd lopen. Daarna wordt er random een paar gekozen waarvan de niet besmette cel besmet wordt. Deze keuze is gewogen op basis van de besmettelijkheid. In het begin is de besmettingsdruk laag en dus de tijd tussen besmettingen groot, naar verloop van tijd neemt de besmettingsdruk toe en wordt tijd tussen besmettingen kleiner, totdat een groot deel van de cellen besmet

is, dan daalt de besmettingsdruk weer en neemt de tijd tussen een besmetting weer toe. Dit proces gaat door totdat alle cellen besmet zijn.

De parameters

Het hierboven beschreven model is geprogrammeerd in C#. De code is aan het eind opgenomen. Vervolgens worden er simulaties gedaan met verschillende initiële fracties besmette cellen en verschillende latente periodes. De resultaten van deze simulaties zijn het percentage besmette cellen afgezet tegen de tijd. Dat ziet er als volgt uit.



Figuur 3: Resultaat simulatie

We kunnen voor de besmettingssnelheid van een buurcel de volgende formule afleiden: $T*(1-p)^c$, T is hierin de tijd, p de kans dat GCV een besmetting blokkeert en c de concentratie. De besmettingssnelheid hangt dus af van de tijd, van de kans op besmetting, ofwel resistentie, en de contratie van GCV. Met behulp van deze formule kan het verband met tussen de resultaten van de simulaties en de experimenten van het UMC vergeleken worden. De waarde van de hierboven beschreven formule kan ingevuld worden in de interpolatie functie van de simulatie data. De waarde die daaruit volgt kan vergeleken worden met het de waarde van de experimentele data met dezelfde concentratie. Het doel hiervan is voor een bepaalde stam een waarde p te vinden die de mate van resistentie tegen GCV aangeeft. De methode die ik daarvoor gebruik is de kleinste kwadraten methode. Bij de kleinste kwadraten methode worden de verschillen tussen een datapunt en het punt dat voorspeldt wordt

door het model gekwadrateerd. Vervolgens worden deze gekwadrateerde verschillen bij elkaar opgeteld en uiteindelijk wordt die som geminimaliseerd.

$$\sum_{i=1}^{n}(f(T*(1-p)^c)-F_i)^2$$

Hierin is f het de interpolatiefunctie en F_i zijn de datapunten van de experimentele data. Door deze som te minimaliseren krijg je een waarde voor T en voor p. Waarbij we geïnteresseert zijn in de p waarde want deze zegt iets over de resistentie van de stam.

Afsluiting

Het lastige van modelleren is dat het slechts een schematische weergave van de werkelijkheid is. Je kan dus niet alle effecten in een proces in het model verwerken. Het model zou dan heel complex worden. Je moet je dus steeds afvragen of een bepaalt element genoeg effect heeft om in het model te moeten worden opgenomen. Een ander lastig punt is het duidelijk krijgen hoe de werkelijkheid in elkaar zit. Je bent daarbij afhankelijk van specialisten. De uitdaging is dan ook om alle voor het model relevante informatie van die specialisten te horen te krijgen. Daarbij komt nog dat ook specialisten niet alles weten, je zult dus zaken moeten aannemen in het model. Verder onderzoek kan helpen om zekerheid te krijgen voor de aannames. Zo kan onderzocht worden met een experiment of het terecht was om voor een vierkantsrooster te kiezen of dat een ander rooster beter voldoet aan de werkelijkheid. Je zou dan direct bij het vullen van de 6-well plaat voordat je GCV toevoegt het mengsel kunnen fixeren en daarvan de fluorescentie meten. Dan kan je zien hoeveel cellen initieel besmet zijn. Als je die waarde gaat vergelijken met de fluorescentie bij een hele hoge concentratie GCV dan is de fluorescentie met hoge GCV concentratie gedeeld door de fluorescentie aan het begin een aanwijzing voor het type rooster. Als bijvoorbeeld die berekening 4 oplevert dan is dat een aanwijzing voor een vierkantsrooster. Het model zou dan daaraan aangepast kunnen worden. Op deze manier kan je het model blijven verfijnen.

Code

Listing 1: Cel.cs

```
using System;

namespace Simulatie
{
    public class Cel
    {
        //Aantal buren 4
        public int[] buren = new int[4];
        public double besmettingstijd;
        public bool besmet;
        double random;
        public int locatie;
        int lengte = Program.lengte;
        int breedte = Program.breedte;
        int aantal = Program.lengte * Program.breedte;
        public Cel(int i, Random rnd)
        {
            locatie = i;
            buren = VindBuren(i);
            random = rnd.NextDouble();
            if (random < 0.01)
            {
                besmet = true;
                Program.InitieleBesmetteCellen.Add(i);
            }
        }
        public void Besmet()
        {
            besmet = true;
            besmettingstijd = Program.tijd;
            Program.BesmetteCellen.Add(locatie);
        }
        public int[] VindBuren(int i)
        {
            if (i - lengte < 0)
```

```
            {
                buren[0] = i - lengte + aantal;
            }
            else
            {
                buren[0] = i - lengte;
            }
            //Onder
            if (i + lengte > aantal - 1)
            {
                buren[1] = i + lengte - aantal;
            }
            else
            {
                buren[1] = i + lengte;
            }
            //Links
            if (i % breedte == 0)
            {
                buren[2] = i + breedte - 1;
            }
            else
            {
                buren[2] = i - 1;
            }
            //Rechts
            if ((i + 1) % breedte == 0)
            {
                buren[3] = i - breedte + 1;
            }
            else
            {
                buren[3] = i + 1;
            }
            return buren;
        }
    }
}
```

Listing 2: System.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.IO;
using System.Globalization;

namespace Simulatie
{

//Vervang IS-lijst door Besmettelijkeparen en
   nietbesmettelijke paren
//nieuwe paren toevoegen aan nietbesmettelijke
//Elke stap kijken of nietbesmettelijk paar besmet
//wordt anders een besmetting

    class Program
    {
    static public int breedte = 500;
    static public int lengte = 500;
    static public double Tijdnietbesmettelijk = 0.001;
    static public int totaal = lengte * breedte;
    public static List<int> BesmetteCellen = new List<int>()
       ;
    public static List<int> InitieleBesmetteCellen = new
       List<int>();
    public static List<Cel> Cellen = new List<Cel>();
    public static double tijd = Tijdnietbesmettelijk;
    static Random rnd = new Random();
    public static List<int[]> BesmetLijst = new List<int
       []>();
    public static List<int[]> NogNietLijst = new List<int
       []>();
    public static List<int>[] Locaties_In_NogNietLijst = new
        List<int>[totaal];
    public static List<int>[] Locaties_In_BesmetLijst = new
       List<int>[totaal];
```

```
public static List<string> TijdAantal = new List<string
   >();
public static int aantal = 0;


    static void Main(string[] args)
    {
        //maak alle cellen
        for (int i = 0; i <= totaal - 1; i++)
        {
            Cellen.Add(new Cel(i, rnd));
            Locaties_In_BesmetLijst[i] = new List<int>()
               ;
            Locaties_In_NogNietLijst[i] = new List<int
               >();
        }
        //De buren van de besmette cellen moeten besmet
           worden
        foreach (int k in InitieleBesmetteCellen)
        {
            BesmetteCellen.Add(k);
            BesmetBuren(k);
        }
        foreach(int x in BesmetteCellen)
        {
            Add_To_BesmetList(Cellen[x]);
            aantal++;
        }
        Simuleer();
        PrintOplossing(TijdAantal, "resultaat.txt");

    }

    static void PrintOplossing(List<string> resultaten,
       string naam)
    {
        File.WriteAllText(naam, "");
        StreamWriter writer = new StreamWriter(naam);
```

```
        writer.AutoFlush = true;
        for (int i =0;i<resultaten.Count();i++)
        {
            writer.WriteLine(resultaten[i]);
        }
    }

    static void Simuleer()
    {
        // nodig lijst met nog niet besmettelijke si
           paren met fifo
        // bij elke tijdstap kijken of eerste si paar
           besmettelijk
        //wordt of dat er een besmetting plaats vindt
        //


        double u;
        //double x;
        double nexttime;
        // initi le besmettingsdruk
        double besmettingsdruk;
        besmettingsdruk = 0;
        int keuze;
        int w = 0;
        double weging = 0;
        TijdAantal.Add("0" + "," + (BesmetteCellen.Count
           ()/(double)totaal*100).ToString(CultureInfo.
           CreateSpecificCulture("en-GB")));
        while (aantal < totaal)
        {
            //exponentieel verdeeld tijd nieuwe
               besmetting
            //lambda = aantalparen;
            bool test = true;
            int aantalparen = BesmetLijst.Count();
            Cel nieuwebesmetting;
            u = rnd.NextDouble();
```

```
//x = Math.Log(1 - u) / (-1 * lambda);
if (aantalparen == 0)
{
    nexttime = 0;
}
else
{
    nexttime = (-besmettingsdruk + Math.Sqrt
       (besmettingsdruk * besmettingsdruk - 2
        * aantalparen * Math.Log(u))) /
       aantalparen;
}
tijd = tijd + nexttime;

if (NogNietLijst.Count() > 0)
{
    int teller = 0;
    while (true)
    {
        if (teller < NogNietLijst.Count() &&
            Cellen[NogNietLijst[teller][1]].
           besmet)
        {
            teller++;
        }
        else
        {
            NogNietLijst.RemoveRange(0,
               teller);
            break;
        }
    }
    if(NogNietLijst.Count() > 0 && Cellen[
       NogNietLijst[0][0]].besmettingstijd +
       Tijdnietbesmettelijk <= tijd)
    {
```

```
            besmettingsdruk = besmettingsdruk +
                BesmetLijst.Count() * (Cellen[
                NogNietLijst[0][0]].
                besmettingstijd +
                Tijdnietbesmettelijk - tijd +
                nexttime);
                BesmetLijst.Add(NogNietLijst[0])
                    ;
                Locaties_In_BesmetLijst[
                    BesmetLijst[BesmetLijst.Count
                    () - 1][1]].Add(BesmetLijst.
                    Count() - 1);
                test = false;
                tijd = Cellen[NogNietLijst
                    [0][0]].besmettingstijd +
                    Tijdnietbesmettelijk;
                NogNietLijst.RemoveAt(0);
        }
    }


    if(BesmetLijst.Count > 0 && test)
    {
        ////Nieuwe besmetting
        //random uniform * besmettingsdruk
        //tijd in doubles
        double random = rnd.NextDouble();
        weging = random * besmettingsdruk;
        double som = 0;
        keuze = 0;
        //bepaal welk paar besmet wordt
        for (int k = 0; k < BesmetLijst.Count();
            k++)
        {
            if (som > weging)
            {
                keuze = k;
                break;
```

```
        }
        else
        {
            som += Functie(Cellen[
                BesmetLijst[k][0]]);
        }
    }
    nieuwebesmetting = Cellen[BesmetLijst[
        keuze][1]];
    nieuwebesmetting.Besmet();
    aantal++;
    for (int j = 0; j <
        Locaties_In_BesmetLijst[
        nieuwebesmetting.locatie].Count(); j
        ++)
    {
        int plaats = Locaties_In_BesmetLijst
            [nieuwebesmetting.locatie][j];
        //plaats laatste paar op gekozen
            plek
        besmettingsdruk = besmettingsdruk -
            (tijd - Cellen[BesmetLijst[keuze
            ][0]].besmettingstijd -
            Tijdnietbesmettelijk);
        int laatst = BesmetLijst.Count() -
            1;
        BesmetLijst[plaats] = BesmetLijst[
            laatst];
        //Pas locatie van het verplaatste
            paar aan in de locatielijst
        if (Locaties_In_BesmetLijst[
            BesmetLijst[plaats][1]] ==
            Locaties_In_BesmetLijst[
            nieuwebesmetting.locatie])
        {
            for (int m = j; m <
                Locaties_In_BesmetLijst[
                BesmetLijst[plaats][1]].Count
```

```
                ();m++)
            {
                if (Locaties_In_BesmetLijst[
                    BesmetLijst[plaats][1]][m]
                    == laatst)
                {
                    Locaties_In_BesmetLijst[
                        BesmetLijst[plaats
                        ][1]][m] = plaats;
                    break;
                }
            }
        }
        else
        {
            for (int m = 0; m <
                Locaties_In_BesmetLijst[
                BesmetLijst[plaats][1]].Count
                (); m++)
            {
                if (Locaties_In_BesmetLijst[
                    BesmetLijst[plaats][1]][m]
                    == laatst)
                {
                    Locaties_In_BesmetLijst[
                        BesmetLijst[plaats
                        ][1]][m] = plaats;
                    break;
                }
            }
        }

        BesmetLijst.RemoveAt(laatst);
    }

    Locaties_In_BesmetLijst[nieuwebesmetting
        .locatie].Clear();
```

```
                Locaties_In_NogNietLijst[
                    nieuwebesmetting.locatie].Clear();
                Add_To_NogNietList(nieuwebesmetting);
                w++;
                double percentage = (BesmetteCellen.
                    Count() / (double)totaal) * 100;
                TijdAantal.Add(tijd.ToString(CultureInfo
                    .CreateSpecificCulture("en-GB")) + ","
                     + percentage.ToString(CultureInfo.
                    CreateSpecificCulture("en-GB")));
                besmettingsdruk = besmettingsdruk +
                    BesmetLijst.Count() * nexttime;
            }

        }
    }
    static void Add_To_NogNietList(Cel cel)
    {

        foreach (int k in cel.buren)
        {
            if (Cellen[k].besmet != true)
            {
                //ISLijst[aantalparen][0] = cel.locatie;
                //ISLijst[aantalparen][1] = k;
                int[] paar = new int[2];
                paar[0] = cel.locatie;
                paar[1] = k;
                NogNietLijst.Add(paar);
                Locaties_In_NogNietLijst[k].Add(
                    NogNietLijst.Count()-1);
            }
        }
    }
    static void Add_To_BesmetList(Cel cel)
    {
        foreach (int k in cel.buren)
        {
```

```
                if (Cellen[k].besmet != true)
                {;
                    int[] paar = new int[2];
                    paar[0] = cel.locatie;
                    paar[1] = k;
                    BesmetLijst.Add(paar);
                    Locaties_In_BesmetLijst[k].Add(
                       BesmetLijst.Count() - 1);
                }
            }
        }
        static void BesmetBuren(int i)
        {
            Cel besmetter = Cellen[i];
            foreach (int buur in besmetter.buren)
            {
                if (!Cellen[buur].besmet)
                {
                    Cellen[buur].Besmet();
                }
            }
        }
        static double Functie(Cel cel)
        {
        //geeft besmettelijkheid van cel
        double w = cel.besmettingstijd;
                double score = tijd - w -
                   Tijdnietbesmettelijk;
                return score;

        }

    }
}
```