BACHELOROPDRACHT

De waterzuivering als Petrinet

Het voorspellen van overstromingen

Loes Knoben
s1010891
Maarten Otten
s0193208
Ruben Fransen
s1004816

21 juni 2013

UNIVERSITEIT TWENTE.

Samenvatting

In dit verslag is gekeken naar de rioolwaterzuiveringsinstallatie van Enschede. Afgelopen jaren is het af en toe voorgekomen dat de installatie niet al het water aankon, waardoor er sprake was van een overstroming. Er is gekeken wat hierbij de risicofactoren van de zuivering zijn.

Om dergelijke vragen te onderzoeken is allereerst de waterzuivering gemodelleerd als een hybride Petrinet. Dit is een wiskundige modelleertaal die gebruikt kan worden om het gedrag van een proces grafisch weer te geven en op een exacte manier te modelleren. Het voordeel van het gebruik van het hybride Petrinet is dat continue en discrete processen gecombineerd kunnen worden tot een zeer groot model en er ook gebruik gemaakt kan worden van stochastische overgangen.

Het basismodel van de waterzuivering bestaat uit vier componenten. Allereerst de aanvoer, waarvan de hoeveelheid wisselt bij wisseling van het weer. Verder is er een buffer waar het water wordt opgeslagen wanneer er teveel binnenkomt om verwerkt te worden. Ook het gemaal dat het water naar binnen pompt is een belangrijk onderdeel, aangezien dit kapot kan gaan. Wanneer dit gebeurt kan geen water meer aangenomen kan worden. Als laatste is het daadwerkelijke zuiveringsproces opgenomen, waar ook de totale hoeveelheid gezuiverd water bijgehouden wordt. Aan dit basismodel kunnen nog uitbreidingen toegevoegd worden, waardoor bijvoorbeeld ook meegenomen kan worden dat slechts een gedeelte van de pompen kapot gaat of dat de totale hoeveelheid overstroomd water bijgehouden wordt.

Het gedrag van dit Petrinet kan bekeken worden door middel van simulatie of analyse. Een analyse omgeving die een kansverdeling voor de inhoud van een bepaalde plaats geeft is reeds beschikbaar, de Fluid Survival Tool (FST). In deze tool kan echter slechts één stochastische overgang meegenomen worden, terwijl het uitgebreide model van de waterzuivering meerdere stochastische onderdelen bevat. Daardoor is een eigen simulatie programma geschreven dat het gedrag van dergelijke modellen kan benaderen. Dit programma in C++ kan uitgebreidere Petrinetten simuleren, maar geeft geen exacte kansverdeling en is in zijn huidige vorm nog niet geschikt om op een makkelijke manier nieuwe Petrinetten in te voeren. Dit laatste is wel het geval bij de bestaande analyse omgeving.

Om de werking van het simulatie programma te testen zijn meerdere situaties vergeleken met beide programma's. Hiervoor is gebruik gemaakt van het basismodel, aangezien deze met beide gemodelleerd kan worden. Allereerst is gekeken naar een deterministische situatie, waaruit blijkt dat de programma's overeen komen. Vervolgens zijn verschillende kansverdelingen geanalyseerd. Een kansverdeling met een grotere variantie blijkt meer runs te vereisen om te convergeren naar de verdeling gegeven door FST. Het programma kan bij het basismodel in alle gevallen binnen een minuut een zodanig groot aantal runs simuleren dat de simulatie voldoet aan de gewenste betrouwbaarheid van 95%. Dit geldt ook wanneer er gebruik wordt gemaakt van meerdere kansverdelingen.
Uit de verificatie blijkt dus dat het programma goed werkt en binnen afzienbare tijd een goede betrouwbaarheid kan bereiken. Dit betekent dat het gebruikt kan worden om Petrinetten met verschillende kansverdelingen te simuleren en hierover uitspraken te doen.

Inhoudsopgave

1 Inleiding

Wanneer het een lange tijd heel hard regent in Enschede, kan de rioolwaterzuivering dit water niet allemaal aan en tunnels in Enschede onder water. Aangezien dit veel overlast veroorzaakt is hierbij van belang welke hoeveelheid regen dit probleem veroorzaakt. Wanneer dit bekend is, kunnen voorzorgsmaatregelen kunnen worden genomen op het moment dat een overstroming dreigt. Verder is het bijvoorbeeld ook interessant om te weten of een terrorist hetzelfde zou kunnen bereiken door een deel van de zuivering te saboteren en welk deel van de installatie hier het meest gevoelig voor is.

Een goede manier om dergelijke grootschalige processen te modelleren is met behulp van Petrinetten. Dit is een wiskundige modelleertaal die gebruikt kan worden om het gedrag te analyseren van een netwerk waarin veel gelijktijdige processen voorkomen. Bovendien kunnen in bepaalde Petrinetten zowel continue, discrete en stochastische processen gecombineerd worden, wat in het geval van de waterzuivering een vereiste is. Met behulp van Petrinetten kunnen de problemen van de waterzuivering geanalyseerd worden en kan gekeken worden wat de beste oplossing is voor bepaalde risicofactoren.

In eerder onderzoek zijn Petrinetten al gebruikt om het gedrag van verschillende soorten complexe netwerken te analyseren[[10]]. Hiervoor is reeds verschillende software beschikbaar. Zo is onder andere een drinkwaterzuivering gemodelleerd [[8]], waarbij de analyse is gedaan met de Fluid Survival Tool [[1]]. De gebruikte software kan exacte analyse uitvoeren op een model met één stochastische component. Bij het modelleren van een rioolwaterzuivering spelen echter meerdere stochastische onderdelen een rol, zoals de verandering van het weer en de reparatie van een pomp. Om deze reden is gekozen een eigen simulatieprogramma te ontwerpen dat meerdere stochastische elementen kan meenemen. Dit programma is niet zo snel en exact als de bestaande tool, maar kan toch binnen een minuut een goede betrouwbaarheid bereiken.

In dit verslag wordt allereerst in hoofdstuk 2 achtergrondinformatie over de werking van de waterzuivering gegeven, met aansluitend de probleemstelling. Vervolgens staat de theorie over Petrinetten in hoofdstuk 3 en wordt in hoofdstuk 4 de waterzuivering als een Petrinet gemodelleerd. In hoofdstuk 5 komt aan bod welke implementatie gebruikt is om het gedrag van Petrinetten te analyseren en hierna wordt deze implementatie gevalideerd en geverifieerd in hoofdstuk 6. Als laatst volgen nog een discussie en conclusie van de resultaten en wordt het verslag afgesloten met aanbevelingen voor verder onderzoek.

2 Probleemomschrijving

Om de rioolwaterzuiveringsinstallatie (rwzi) Enschede te kunnen modelleren is het nodig de werking van een waterzuivering te begrijpen en te weten welke problemen een rol kunnen spelen. Om dit te bereiken is onderzoek op de locatie gedaan en heeft een gesprek plaats gevonden met een werknemer van de installatie. In dit hoofdstuk wordt nader ingegaan op het zuiveringsproces en de specifieke risicofactoren van de zuivering in Enschede.

2.1 Werking rioolwaterzuiveringsinstallatie

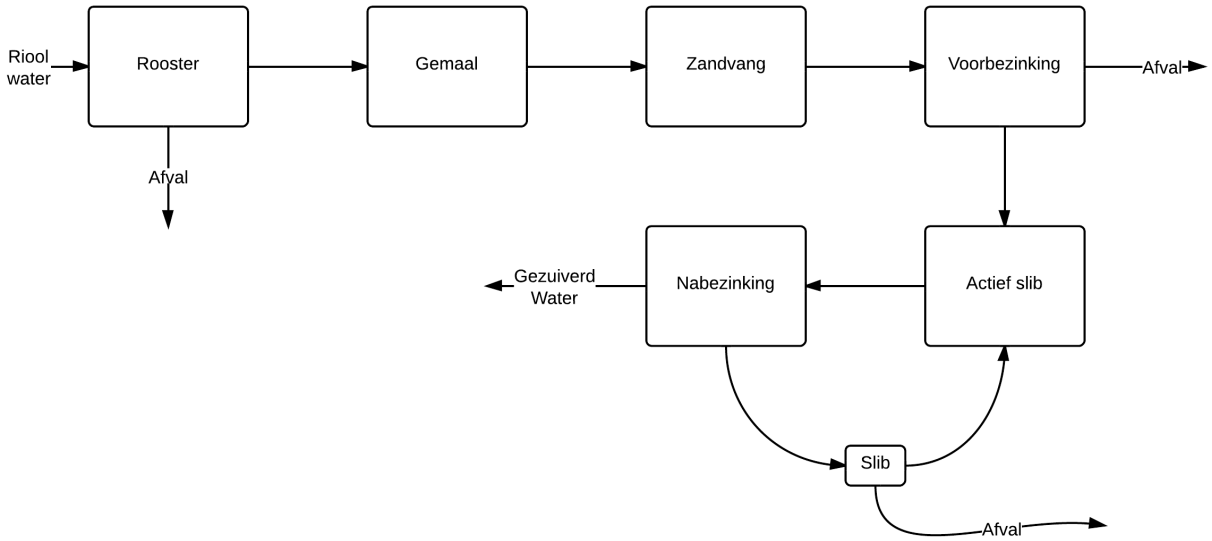
In een rioolwaterzuivering wordt afvalwater uit het riool zodanig gezuiverd dat het zonder schadelijke gevolgen terug kan vloeien in het oppervlaktewater. Het zuiveringsproces bestaat uit verschillende stappen.

Het water uit het riool komt door aanvoervijzelgemalen aan bij de waterzuivering. Allereerst gaat het door een roosterhark, waar de grove bestanddelen uit het water worden verwijderd. Dit zijn bijvoorbeeld stukken hout en plastic die de installatie zouden kunnen verstoppen. Hierna wordt het water door een vijzelgemaal omhoog gepompt. Het laatste onderdeel voor de echte zuivering is de zandvang. Hier wordt het zand uit het water verwijderd zodat het geen schade kan aanrichten aan de installatieonderdelen. Door het water in een grote bak stil te laten staan, zinkt het zand naar de bodem. Het water dat aan de bovenkant over de bak heen stroomt is daardoor grotendeels vrij van zand. Hierna wordt het water verdeeld over 3 zuiveringsstraten met eenieder nog een eigen vijzelgemaal om het water op genoeg hoogte te brengen en het de rest van het proces in vrijverval te laten doorstromen.

Het echte proces begint met de voorbezinktanks. Hierin staat het water lang genoeg stil om ook het fijnere slib naar de bodem te laten zakken. Dit vrijgekomen slib wordt vergist. Bij het vergisten komt biogas vrij, wat door de zuivering gebruikt wordt om eigen energie op te wekken. Het resterende, niet vergistte, slib wordt verbrand.

Na de voorbezinktanks gaat het water door de actief-slibinstallatie. Hier wordt actief slib met geschikte bacteriën erin gebruikt om stikstof en fosfaat uit het water te halen, deze bacteriën gebruiken de verontreinigingen in het water namelijk als voedsel. Daarnaast moet er in deze stap nog zuurstof aan het water toegevoegd worden om deze bacteriën in leven te houden. Als laatste gaat het water met het actieve slib erin naar de nabezinktank. Het slib bezinkt hier. Een deel hiervan kan opnieuw gebruikt worden in de actief slib-installatie, de rest wordt op dezelfde manier verwerkt als het slib uit de voorbezinktanks. Het schone water stroomt aan de bovenkant de tank uit en kan dan de sloot in. Een schematische weergave hiervan is te zien in figuur 1.
De maximale hoeveelheid water die uiteindelijk per uur gezuiverd kan worden hangt af van de capaciteit van de aanvoervijzelgemalen. Deze capaciteit is zodanig afgestemd dat bij maximale aanvoer het water lang genoeg verblijft in alle tanks om het gezuiverde water aan de eisen te laten voldoen. Wanneer er sprake is van een lage aanvoer zullen de gemalen minder water aanvoeren en zal het water daardoor in bepaalde tanks langer stilstaan.

Gegevens installatie in Enschede In de rioolwaterzuiveringsinstallatie Enschede komt afvalwater uit Enschede, Boekelo, Lonneker en Usselo binnen. Het water uit Enschede wordt door 4 aanvoervijzelgemalen naar de installatie gepompt. Deze vijzels hebben allemaal een capaciteit van 2.390 m^3/h, dat is samen een totale capaciteit van 12.160 m^3/h. Het zuiveren begint bij



Figuur 1: Schematische weergave rioolwaterzuiveringsinstallatie.

het verwijderen van het grove vuil doormiddel van roosterharken en het water stroomt daarna na de zandvang. Nadat het water door een zandvang is geweest wordt het door een verdeelwerk opgesplitst in 3 verschillende 'straten'. Door 3 processen gescheiden te houden kan eventueel een straat worden afgesloten wanneer er bijvoorbeeld een extra giftige lading water binnenkomt. In dit geval worden slechts in één of twee bakken de bacteriën aangetast. De eerste straat heeft een capaciteit van 5.910 m^3/h en de tweede en derde straat hebben samen een capaciteit van 6.250 m^3/h. Bij al deze straten is een tussengemaal aanwezig wat het water omhoog pompt zodat het daarna onder vrij verval door de voorbezinktanks, de actief-slibinstallaties en nabezinktanks kan gaan. Uiteindelijk belandt het schone water in de Elsbeek.
De gemiddelde droogweer-aanvoer van rwzi Enschede is 3.240 m^3/h. De hoeveelheid water voor de vijzels wordt continu gemeten en de capaciteit van de vijzels wordt hieraan aangepast. Dit betekent dat bij droog weer niet alle aanvoervijzels in gebruik zijn. Alle 'straten' achter de vijzels blijven in dit geval wel in werking. De maximale water aanvoer is 12.160 m^3/h. Wanneer er meer afvalwater aankomt wordt de toevoer afgesloten en is het vervolgens de taak van de gemeente om het overtollige water op te slaan totdat dit weer aangenomen kan worden. Om te voorkomen dat er een overstroming plaats vindt heeft de gemeente hiervoor buffers aangelegd, die langzaam vollopen bij een hoge aanvoer. Deze buffers zijn echter niet oneindig groot en is er bij extreme regenval een kans op overstroming.

2.2 Risicofactoren installatie Enschede

- Wanneer het heel hard regent is er meer afvalwater dan er aangenomen kan worden. Wanneer de buffers niet genoeg capaciteit hebben om dit aan te nemen is er kans op overstroming.
- Wanneer er één of meerdere aanvoervijzelgemalen stuk gaan kan er minder of geen water aangenomen worden van de gemeente.

- Wanneer een van de tussengemalen bij de zuiveringsstraten stuk gaat komen één of meer straten volledig stil te liggen en kan er minder of geen water verwerkt worden.
- Wanneer de stroom uitvalt vallen alle gemalen uit en kan er helemaal geen water meer aangenomen worden. Wanneer dit lang duurt kan er in overleg een noodaggregaat ingeschakeld worden. Deze is echter niet direct op het terrein beschikbaar.
- Wanneer er een giftige lading water komt, gaat een deel van de bacteriën dood en moeten er extra bacteriën worden toegevoegd of deze moeten de tijd hebben om te herstellen.
- Wanneer de temperatuur onder de 5°C komt zijn de bacteriën minder actief, waardoor er extra zuurstof toegevoegd moet worden.
- Wanneer het een tijd niet geregend heeft komt bij de eerstvolgende regenbui extra veel slib mee. Hierdoor kan er minder water aangenomen worden, omdat anders niet al het slib gefilterd kan worden. Bovendien kan de grote hoeveelheid gas die vrijkomt bij de vergisting van het slib niet allemaal omgezet worden tot elektriciteit. Een deel van het gas wordt dan afgefakkeld in plaats van gebruikt.

2.3 Probleemstelling

We zijn geïnteresseerd in het in kaart brengen van de problemen en risicofactoren van de waterzuivering. Op deze manier kan gekeken worden waar de zwakke punten zitten en hoe deze verbeterd kunnen worden. Om de waterzuivering wiskundig goed te kunnen analyseren moet het allereerst in een wiskundig model worden gegoten. We doen dit door middel van een Petrinet, aangezien het gedrag van de waterzuivering hiermee op een duidelijke manier in kaart gebracht kan worden. De eerste stap zal dus het vertalen van de praktijk van de waterzuivering naar het model van het Petrinet zijn.

Vervolgens willen we bekijken hoe vaak bepaalde toestanden van het Petrinet zich voordoen. Een toestand is in dit geval bijvoorbeeld het overstromen van de buffer of het uitvallen van een gemaal. Om hierover uitspraken te doen, moet het Petrinet gesimuleerd of geanalyseerd kunnen worden. De belangrijkste doelstelling is daarom ook de ontwikkeling van een programma dat een Petrinet kan simuleren om zo de benodigde kansen te verkrijgen. Daarnaast is het natuurlijk belangrijk om de werking van dit programma te controleren om te zien of het ook daadwerkelijk gebruikt kan worden om Petrinetten correct te simuleren.

Met behulp van bovenstaande omgeving is het uiteindelijk mogelijk het gedrag van de waterzuivering te analyseren. Vervolgens kan worden bepaald welke van de eerdergenoemde risicofactoren ook daadwerkelijk problemen kunnen opleveren.

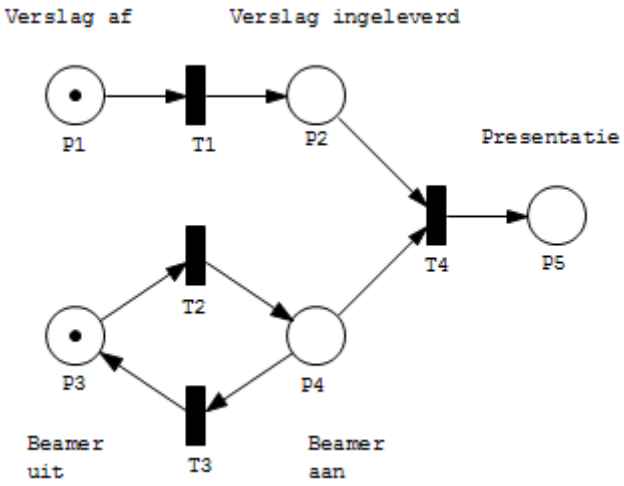
3 Theorie Petrinetten

Een Petrinet is een wiskundige modelleertaal die gebruikt kan worden om het gedrag van een proces met veel componenten en onderdelen te beschrijven. Een bijkomend voordeel van een Petrinet is dat het naast een precieze wiskundige beschrijving, ook een duidelijke grafische weergave van het proces geeft. Aangezien het standaard Petrinet erg algemeen en flexibel is, zijn er vele varianten en uitbreidingen, elk gericht op hun eigen toepassing. Hierdoor is het model in veel verschillende vakgebieden bruikbaar, bijvoorbeeld in industriële netwerken en biologische netwerken. [10]

3.1 Standaard Petrinet

Een Petrinet is een gerichte bipartite graaf die bestaat uit twee soorten knopen, namelijk plaatsen en transities. Deze plaatsen en transities vormen de bipartitie, de pijlen verbinden alleen plaatsen met transities en omgekeerd. De plaatsen in een Petrinet worden weergegeven als cirkels en kunnen een bepaald aantal tokens bevatten. Het aantal tokens in iedere plaats geeft de toestand waarin het Petrinet zich bevindt. Deze tokens bewegen zich door het Petrinet via de transities, die weergegeven worden als rechthoeken of vierkanten. Wanneer een transitie vuurt worden één of meerdere tokens verplaatst, waardoor de toestand van het Petrinet verandert.[11], [12]

Een voorbeeld van een situatie die te modelleren is als een Petrinet is het geven van de eindpresentatie van de bacheloropdracht. Om deze presentatie te kunnen houden moet op de dag van de presentatie aan een aantal voorwaarden zijn voldaan. Allereerst moet het verslag ingeleverd zijn, maar om dit te bereiken moet om te beginnen het verslag af zijn. Daarnaast is er nog een andere voorwaarde, ook de beamer moet aan staan. Wanneer dit niet het geval is, zal de beamer eerst aangezet moeten worden.



Figuur 2: Het Petrinet bij het geven van de eindpresentatie. Om hiervoor in de goede toestand te zijn moet eerst de beamer aan zijn en het verslag ingeleverd zijn.

Deze situatie is te modelleren als een Petrinet, dat te zien is in figuur 2. De plaats van de tokens geeft weer in welke toestand het systeem zich bevindt. Wanneer in $\mathcal{P}_1$ een token is betekent dit dat het verslag af is. Verder geeft een token in $\mathcal{P}_2$ dat het verslag ook ingeleverd is. Daarnaast betekent een token in $\mathcal{P}_3$ dat de beamer uitstaat en een token in $\mathcal{P}_4$ dat de beamer aan staat. Als laatste vindt de presentatie plaats wanneer een token in $\mathcal{P}_5$ is.
De transities kunnen deze toestanden veranderen door tokens te verplaatsen. Zo zal transitie $\mathcal{T}_1$ vuren op de dag dat het verslag ingeleverd moet worden, waarbij het token verplaatst wordt zodat het systeem zich in de toestand bevindt waarbij het verslag ingeleverd is. Dit kan echter alleen plaatsvinden wanneer er een token in $\mathcal{P}_1$ aanwezig is. Verder stelt transitie $\mathcal{T}_2$ de aanknop van de beamer voor. Wanneer hierop gedrukt wordt zal deze transitie het token verplaatsen naar de toestand waarin de beamer aanstaat. Op dezelfde manier correspondeert transitie $\mathcal{T}_3$ met de uitknop van de beamer. Als laatste is er de transitie $\mathcal{T}_4$, die is ingesteld om te vuren op het moment dat de presentatie is gepland, dit kan echter alleen plaatsvinden wanneer er zowel in $\mathcal{P}_2$ als in $\mathcal{P}_4$ een token is. Wanneer dit het geval is zullen deze tokens samengevoegd worden tot één token in plaats $\mathcal{P}_5$.
Om dus de eindpresentatie te kunnen geven moet deze laatste transitie vuren om het systeem in de goede toestand te krijgen. Er is in het Petrinet te zien dat hiervoor eerst het verslag ingeleverd moet zijn door de eerste transitie te laten vuren. Bovendien moet met behulp van de aanknop de beamer aangezet zijn.

Notatie Formeel wordt een Petrinet beschreven door het tuple $PN = (N, M, W)$. Hierin is $N = (\mathcal{P}, \mathcal{T}, \mathcal{A})$ een net met $\mathcal{P}$ de verzameling plaatsen, $\mathcal{T}$ de verzameling transities en $\mathcal{A} \subset (\mathcal{P} \times \mathcal{T}) \cup (\mathcal{T} \times \mathcal{P})$ de verzameling pijlen. De verzameling pijlen is dus een deelverzameling van alle mogelijke verbindingen tussen alle plaatsen en alle transities en omgekeerd. Verder geldt dat $M : \mathcal{P} \to \mathbb{Z}$ de markering is, deze geeft het aantal tokens per plaats in de begintoestand. Daarnaast geeft $W : \mathcal{A} \to \mathbb{Z}$ het gewicht van een pijl, dit is het aantal tokens dat bij het vuren van een transitie over een pijl gaat. [6]

De toestand van een Petrinet verandert dus door het vuren van de transities, een transitie T kan echter alleen vuren als aan alle voorwaarden is voldaan. Dit is het geval wanneer iedere input plaats P_i van deze transitie een marking M_i heeft van minstens W_i tokens, waarbij W_i het gewicht is van de pijl A_i van P_i naar T. Wanneer niet specifiek een gewicht is gegeven, geldt het standaardgewicht van één. Op het moment dat de transitie daadwerkelijk vuurt, verwijdert hij W_i tokens uit input plaats P_i en voegt hij W_o tokens toe aan de output plaats P_o. [6]

3.2 Hybride Petrinet

In een standaard Petrinet zijn de plaatsen en transities discreet, er kan slechts een geheel aantal tokens verplaatst worden. Een groot deel van de processen in de waterzuivering is echter continu. De hoeveelheid water die door een transitie wordt verplaatst kan bijvoorbeeld niet uitgedrukt worden in tokens, dit is een willekeurige hoeveelheid. De werking van een pomp in de zuivering is echter wel discreet, hij is aan of uit. Om de waterzuivering te modelleren met een Petrinet is er daarom een uitbreiding nodig die discrete en continue processen kan combineren. Er is daarom gekozen om het hybride Petrinet hiervoor te gebruiken.

Aangezien een hybride Petrinet een uitbreiding is van een gewoon Petrinet bevat het ook plaatsen, transities en pijlen, echter zijn er meerdere soorten. Naast de gewone, discrete plaatsen zijn er continue plaatsen, die in plaats van een geheel aantal tokens een willekeurige hoeveelheid 'vloeistof' bevatten. Verder zijn er in totaal vier soorten transities. Drie van deze transities zijn discreet en verplaatsen bij het vuren een aantal tokens. Onmiddellijke transities vuren hierbij direct nadat ze geactiveerd zijn, deterministische transities vuren na een vaste tijd vanaf het moment dat ze geactiveerd zijn en stochastische transities vuren met een bepaalde kansverdeling vanaf het moment dat ze geactiveerd zijn. Daarnaast zijn er nog continue transities. Deze verplaatsen geen tokens, maar hebben een stroomsnelheid en verplaatsen een variabele hoeveelheid van de ene naar de andere continue plaats. [4]

Bij de waterzuivering is de buffer een voorbeeld van een continue plaats waarin water opgeslagen kan worden. Een discrete plaats kan hier gebruikt worden om de werking van een pomp aan te geven, de pomp is aan of uit. Voorbeelden van continue transities zijn de aanvoer die binnenkomt of het water dat van het ene naar het andere bassin stroomt, deze hebben namelijk een bepaalde stroomsnelheid. Bij een deterministische transitie kan gedacht worden aan het moment waarop een bepaalde pomp stuk gaat, terwijl een stochastische transitie vaak een onbekende reparatietijd weergeeft.

Notatie Formeel wordt een hybride Petrinet beschreven door het tuple

$$HPN = (\mathcal{P}, \mathcal{T}, \mathcal{A}, M, \Phi).$$

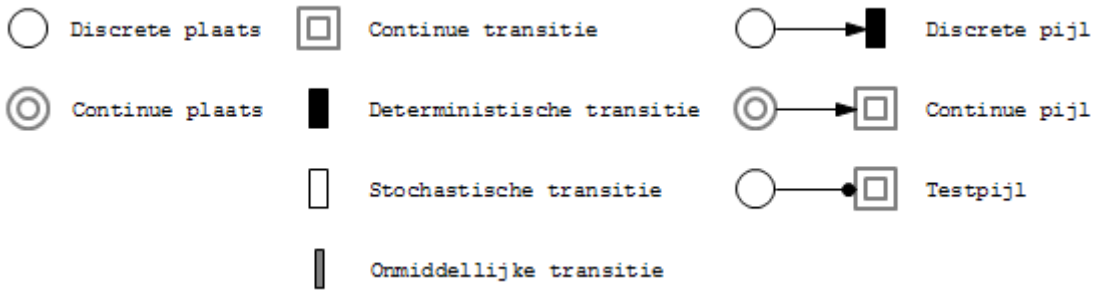
Hierin is $\mathcal{P}$ weer de verzameling plaatsen, alleen bestaat deze nu uit discrete en continue plaatsen, $\mathcal{P} = \mathcal{P}_D \cup \mathcal{P}_C$. De verzameling transities bestaat in een hybride Petrinet uit vier soorten transities. Continue transities ($\mathcal{T}_C$) zijn verbonden met continue plaatsen, daarnaast zijn er tussen discrete plaatsen onmiddellijke transities ($\mathcal{T}_I$), deterministische transities ($\mathcal{T}_D$) en stochastische transities($\mathcal{T}_S$) mogelijk. Dus $\mathcal{T} = \mathcal{T}_C \cup \mathcal{T}_I \cup \mathcal{T}_D \cup \mathcal{T}_S$.
De verzameling $\mathcal{A} = \mathcal{A}_D \cup \mathcal{A}_C \cup \mathcal{A}_T$ van pijlen bestaat uit pijlen die met discrete plaatsen zijn verbonden $\mathcal{A}_D \subset (\mathcal{P}_D \times \mathcal{T}_D) \cup (\mathcal{T}_D \times \mathcal{P}_D)$, pijlen die met continue plaatsen zijn verbonden, $\mathcal{A}_C \subset (\mathcal{P}_C \times \mathcal{T}_C) \cup (\mathcal{T}_C \times \mathcal{P}_C)$ en test pijlen die discrete plaatsen met transities verbinden, $\mathcal{A}_T \subset \mathcal{P}_D \times \mathcal{T}$. Testpijlen zijn de enige pijlen die niets verplaatsen, zij testen slechts of een bepaald aantal tokens in een plaats aanwezig is. In figuur 3 is te zien hoe deze plaatsen, transities en pijlen grafisch weergegeven kunnen worden.

De begintoestand, ook wel de initiële markering van het hybride Petrinet, wordt gegeven door M. Deze bevat van iedere discrete plaats het geheel aantal tokens dat er in de begintoestand aanwezig is en voor iedere continue plaats de willekeurige hoeveelheid op dat moment.

Tenslotte is er het tuple Φ dat bestaat uit negen functies

$$\Phi = (\phi_b^P, \phi_w^T, \phi_p^T, \phi_d^T, \phi_f^T, \phi_g^T, \phi_w^A, \phi_s^A, \phi_p^A),$$

die informatie bevatten over verschillende eigenschappen van de plaatsen, transities en pijlen in het Petrinet. Zo geeft ϕ_b^P voor elke continue plaats een bovengrens voor de maximale inhoud. Verder hebben alle continue transities een constante maximale stroomsnelheid die gegeven wordt door ϕ_f^T. De overige functies zullen hieronder bij hun toepassing worden besproken. [8], [5]



Figuur 3: Grafische weergave van plaatsen, transities en pijlen in een hybride Petrinet.

Vuren van transities Wanneer een transitie vuurt, is voor iedere soort transitie verschillend. Een aantal voorwaarden geldt hierbij voor alle transities. Iedere pijl die vanuit een bepaalde plaats de transitie binnenkomt heeft een gewicht ϕ_w^A. Dit is het aantal tokens of de hoeveelheid die de pijl nodig heeft om deze transitie te kunnen laten vuren. Wanneer er geen gewicht is gegeven heeft de pijl het standaardgewicht van één. De transitie kan pas vuren wanneer aan de voorwaarden van alle binnenkomende pijlen is voldaan. Dit betekent dat voor alle plaatsen die met een pijl naar de transitie toe verbonden zijn, de markering groter of gelijk aan het gewicht van bijbehorende pijl moet zijn. In ons model kan bijvoorbeeld een pomp alleen werken als de stroom aan is.
Wanneer aan bovenstaande voorwaarden is voldaan kan een onmiddellijke transitie direct vuren. Voor een deterministische transitie geldt dat deze vanaf dit moment vuurt met een vertraging. Hierbij is ϕ_d^T de tijd voordat de transitie plaatsvindt vanaf activatie. Bij stochastische transities geeft de functie ϕ_g^T een continue kansverdeling waarna de de transitie vuurt. Naast bovenstaande voorwaarden geldt voor continue transities nog dat minstens één van voorgaande continue plaatsen een inhoud of instroom groter dan nul moet hebben, anders kan er niets verplaatst worden. [7]

Conflictsituaties Het kan zijn dat er conflictsituaties ontstaan, waarin het niet duidelijk is welke transitie er gaat vuren. Bij onmiddelijke en deterministische transties geven daarom ϕ_w^T het gewicht en ϕ_p^T de prioriteit van de transities. In het geval dat meerdere transities tegelijk zouden moeten vuren, vuren allereerst alleen de transities met de hoogste prioriteit. Wanneer er dan nog steeds meerdere transities zijn met eenzelfde prioriteit worden met behulp van de gewichten van de transities voor iedere transitie relatieve kansen uitgerekend die bepalen in welke volgorde het vuren plaatsvindt.

Conflictsituaties tussen continue transities kunnen op een dergelijke manier opgelost worden. In dit geval hebben de continue pijlen naar de transities toe een aandeel ϕ_s^A en een prioriteit ϕ_p^A. Wanneer er meerdere transities zijn die kunnen vuren, wordt allereerst gekeken naar prioriteit. Vervolgens wordt de hoeveelheid die nog over is verdeeld naar aandeel en gewicht. Het kan hierbij dus zijn dat alle transities onder hun normale stroomsnelheid vuren. [7]

3.3 Voordelen Petrinet

- Het model kan op een eenvoudige manier aangepast worden wanneer je verschillende situaties wilt bekijken. Het aantal tokens in een plaats of de tijd waarna een deterministische transitie vuurt kunnen bijvoorbeeld gemakkelijk veranderd worden.
- Voor stochastische transities is het mogelijk om verschillende soorten kansverdelingen te gebruiken. De overgang tussen verschillende toestanden is daardoor in kansverdelingen uit te drukken.
- Het proces is op een duidelijke manier te observeren door de grafische weergave, je kunt bijvoorbeeld het aantal tokens door het net zien bewegen.
- Ook grote processen waarin veel onderdelen parallel lopen zijn te modelleren in een Petrinet.
- Het model is geschikt voor processen waar zowel continue als discrete veranderingen een rol spelen.
- Aangezien het model heel flexibel is zijn er veel verschillende soorten Petrinetten en kunnen er ook nog meer uitbreidingen aan toegevoegd worden. Hierdoor zijn Petrinetten ook inzetbaar bij heel diverse toepassingen.
- Er is uiteenlopende software beschikbaar om het gedrag van een Petrinet mee te bekijken.

3.4 Discrete Event Simulation

Met Petrinetten kunnen ook erg grote systemen beschreven worden. Een dergelijk groot netwerk kan veel transities en plaatsen bevatten en daardoor zo'n grote toestandsruimte omvatten dat deze niet meer direct te analyseren is. Een methode om een dergelijk groot Petrinet te modelleren is dan door middel van Discrete Event Simulation. [3]

Discrete Event Simulation bestaat uit met een kansverdeling gegenereerde initiële condities die een proces doorlopen. Deze manier van simuleren gaat ervan uit dat een proces over een gekozen tijdsperiode, de simulatietijd, verdeeld is in intervallen door een eindig aantal events. Hierbij is van elk interval bekend hoe het proces zich gedraagt. De events zijn dan momenten in de tijdsperiode waarop een verandering van het proces plaatsvindt, waarna de doorloop van het proces tot het volgende event bekend is. Het tijdstip van een event kan deterministisch of stochastisch zijn. Aan het begin worden alle events gegenereerd en in volgorde afgehandeld. Na ieder event wordt gekeken of er nog nieuwe events moeten worden gepland en zo nodig worden deze toegevoegd. In het geval van de waterzuivering zijn voorbeelden van events verandering van de instroom, een storing, en het leeg raken van een bassin.

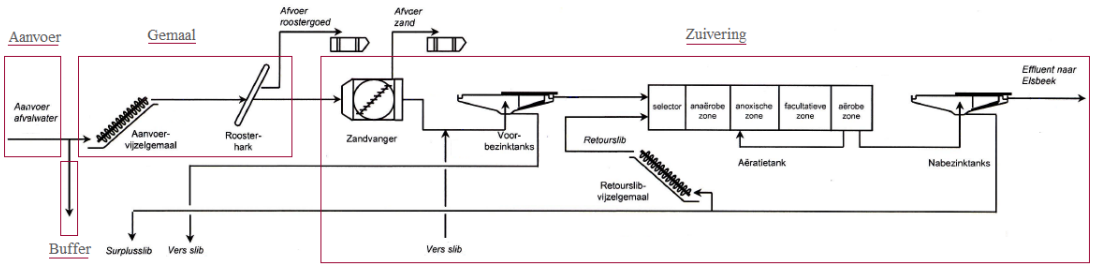
Een run van een Discrete Event Simulation van een Petrinet doorloopt een deel van de mogelijke toestandsruimte waar het Petrinet zich in het gesimuleerde tijdsinterval kan bevinden. Door het draaien van een groot aantal runs kan de toestandsruimte worden benaderd en kan er ook een verwachtingswaarde voor het bereiken van een bepaalde toestand benaderd worden. Zo kunnen er met zekere betrouwbaarheid uitspraken gedaan worden over de bereikbaarheid van toestanden. Voor de waterzuiveringsinstallatie is het bijvoorbeeld interessant om de bereikbaarheid te weten, hoe waarschijnlijk het is dat de buffer overstroomt.

4 Modelformulering

In dit hoofdstuk wordt het wiskundig model van de waterzuivering geformuleerd en wordt toegelicht welke aannames hiervoor gedaan zijn. Allereerst komt het basismodel aan bod en daarna worden nog enige uitbreidingen voorgesteld.

4.1 Basismodel

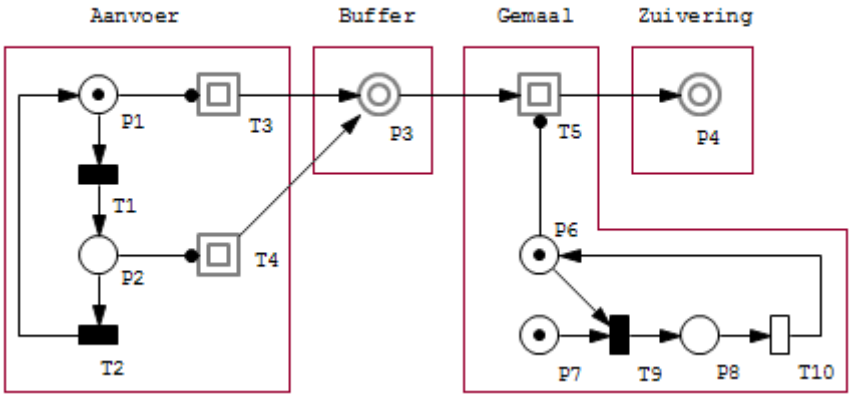
Om het proces van de waterzuivering uitgebreid te kunnen simuleren en analyseren moet het eerst omgezet worden in een wiskundig model. Aangezien er in het proces zowel continue als discrete processen een rol spelen, is ervoor gekozen de waterzuivering weer te geven als een hybride Petrinet. Om de vertaalslag naar een Petrinet te maken is gebruik gemaakt van de geschematiseerde versie van de waterzuivering in Enschede. Het proces is hier opgedeeld in 4 verschillende delen, die apart bekeken worden. Hierbij zijn aannames gedaan om het proces versimpeld weer te geven.



Figuur 4: Basismodel waterzuivering.

Allereerst wordt de aanvoer als een apart stuk bekeken. Dit omdat meegenomen moet worden dat de stroomsnelheid wisselt bij wisseling van het weer. Het water gaat vervolgens naar het gemaal, maar als het hier niet aangenomen kan worden zal het worden opgeslagen in de buffer. Dit is te modelleren door te veronderstellen dat het water altijd door de buffer stroomt en hier blijft staan wanneer het niet aangenomen kan worden. Bij het bekijken van het gemaal moet er rekening mee gehouden worden dat deze kapot kan gaan, waarna er geen water meer door de rest van de installatie gaat. Het laatste deel is het daadwerkelijke zuiveringsproces, waar het water door alle tanks stroomt. Hier geldt dat de tanks volgens het 'overstroom' principe werken en dat het water er onder vrij verloop doorheen stroomt. Hierdoor is de hoeveelheid water die uit elke tank stroomt gelijk en kan dit stuk als een geheel bekeken worden.
Wanneer alle componenten worden omgezet tot een Petrinet, ontstaat het basismodel in figuur 5. Hieronder wordt toegelicht hoe ieder gedeelte is gemodelleerd en wat alle plaatsen en transities voorstellen.

Aanvoer Allereerst beschouwen we de invoerstromen uit verschillende plaatsen als één totale invoerstroom. Aangezien we ervan uit gaan dat het overal gemiddeld even hard regent, maakt het voor de hoeveelheid water die het systeem binnenkomt niet uit waar het precies vandaan komt. Wat wel wisselt is de totale hoeveelheid water die per uur binnenkomt, deze is afhankelijk van het weer. Hierbij is onderscheid gemaakt tussen droog weer en periodes met regen. Dit is



Figuur 5: Basismodel waterzuivering, bron afbeelding RWZI Enschede

gemodelleerd door een token dat verspringt tussen twee discrete plaatsen, waardoor er afwisselend sprake is van droog weer en regen. Transitie $\mathcal{T}_3$ geeft de aanvoer bij droog weer. Hiervoor test de transitie of er een token is in $\mathcal{P}_1$. Wanneer dit het geval is betekent dit dat transitie $\mathcal{T}_3$ kan vuren en $\mathcal{T}_4$ niet. De hoeveelheid water die dan in de buffer binnenkomt is gelijk aan de stroomsnelheid van $\mathcal{T}_3$. Als het begint met regenen verspringt het token naar $\mathcal{P}_2$. Wanneer het token zich hier bevindt is de situatie omgekeerd. Transitie $\mathcal{T}_4$ kan dan vuren, wat de aanvoer bij regen geeft. Verder geeft de deterministische transitie $\mathcal{T}_1$ aan hoe lang het duurt voordat het gaat regenen en bepaalt $\mathcal{T}_2$ hoe lang het hierna duurt voordat het weer droog is.

Buffer Wanneer er meer water aankomt dan de waterzuivering aankan, wordt dit extra water niet aangenomen door de installatie. De toevoer wordt dan stopgezet en de buffer van de gemeente Enschede stroomt vol. De buffer is gemodelleerd als plaats $\mathcal{P}_3$ waar het water doorheen stroomt voordat het de waterzuivering binnengaat. Wanneer er geen water meer wordt toegelaten in de zuivering zal het water hier opgeslagen worden. De bovengrens van deze plaats is gelijk aan de capaciteit van de buffer. Wanneer deze overschreden wordt zal een overstroming plaats vinden.

Gemaal Het aanvoervijzelgemaal ($\mathcal{T}_5$) pompt het water omhoog, deze regelt hierdoor de watertoevoer van de waterzuivering zelf. Het rooster is ook opgenomen in transitie $\mathcal{T}_5$, aangezien het water hier niet stil hoeft te staan. Nadat het naar binnen gepompt is stroomt het namelijk constant doorheen.
Ook moet meegenomen worden dat het gemaal dat de aanvoer regelt kapot kan gaan. De werking hiervan is daarom gemodelleerd als een token in de discrete plaats $\mathcal{P}_6$. Om te kijken wat het effect is van bijvoorbeeld een stroomstoring kan met deterministische transitie $\mathcal{T}_9$ aangegeven worden wanneer het gemaal kapot gaat. Vervolgens geeft $\mathcal{T}_{10}$ de stochastische reparatieduur aan. Het token in de plaats $\mathcal{P}_7$ geeft aan dat het gemaal in een analyse slechts één keer kapot kan gaan. Hierna is er geen token meer in $\mathcal{P}_7$ en kan $\mathcal{T}_9$ dus niet meer vuren.

Zuivering Na het gemaal gaat het water allereerst door de zandvang. Aangezien de zandvang een bassin is dat altijd helemaal vol staat en het water er uit loopt door te 'overstromen', is

de uitstroomsnelheid hiervan altijd gelijk aan de instroomsnelheid. Het is daarom niet nodig hiervoor een aparte plaats of transitie te maken. Hierna begint de rest van het zuiveringsproces, wat volgens hetzelfde principe verloopt. Hierbij is aangenomen dat de tussengemalen geen invloed hebben op de doorstroming. Een hoeveelheid water gaat ook door de voorbezinking, de actief-slibinstallatie en de nabezinking met dezelfde snelheid als het binnenkomt. Verder is in de uitstroom uiteindelijk geen verschil te zien tussen het water uit de verschillende 'straten'. Het gehele zuiveringsproces, inclusief de zandvang, kan daarom weergegeven worden als één continue plaats, $\mathcal{P}_4$. Deze plaats geeft ook meteen de totale uitstroom van het proces, die gelijk is aan de totale hoeveelheid gezuiverd water. De bovengrens van deze plaats is zo hoog als nodig is om al het gezuiverde water aan te kunnen, aangezien er in principe oneindig veel water in de sloot geloosd kan worden. Hiervoor wordt daarom de capaciteit vermenigvuldigd met de simulatietijd genomen, aangezien dit de maximale hoeveelheid water die gezuiverd kan worden.

4.2 Uitbreiding model

Het basismodel uit figuur 5 kan nog niet alle problemen modelleren waarin we geïnteresseerd zijn. Het kan bijvoorbeeld niet bijhouden hoeveel water er overstroomt of wat er gebeurt wanneer er slechts een deel van de pompen kapot gaat. Ook maakt het geen onderscheid tussen een enorme stortbui en gemiddelde regenval. Om ook deze factoren in het model op te nemen zijn er enkele uitbreidingen nodig op het basismodel, die hieronder worden toegelicht.

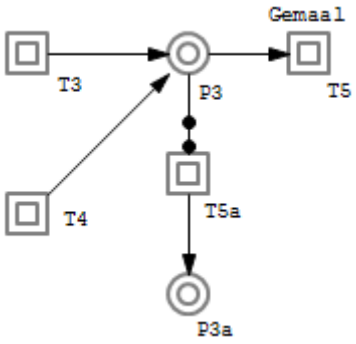
4.2.1 Opslag overstroomwater

In de basis definitie van het petrinet voor de waterzuivering geldt dat een continue plaats nooit kan overstromen. Wanneer een continue plaats zijn bovengrens bereikt en de instroom groter is dan de uitstroom, wordt de instroom automatisch naar beneden bijgesteld. Het vollopen van de buffer zou dan tot gevolg hebben dat het Petrinet de instroom van de regen naar beneden bijstelt. De hoeveelheid regen is in werkelijkheid echter niet zo makkelijk te beheersen. Logischerwijs zou de buffer moeten overstromen. Met name dit moment, dat een plaats zoals de buffer dreigt te overstromen, is hier interessant. Het registreren wanneer dit gebeurt, hoevaak deze plaats overstroomt en hoeveel water hierbij overstroomt is de uitdaging.

Om toch bij te houden hoeveel het overschot aan instroom is, is een uitbreiding van het model nodig. De verzameling pijlen wordt hiervoor uitgebreid met een overstroompijl $\mathcal{A}_o \subset (\mathcal{P}_\mathcal{C} \times \mathcal{T})$. Deze pijl loopt van een continue plaats naar een continue transitie. Hij wordt geactiveerd wanneer de hoeveelheid vloeistof in deze plaats gelijk is aan de testhoeveelheid. De stroomsnelheid van de transitie $\mathcal{T}_5$ wordt dan zodanig aangepast dat het vloeistofniveau in de plaats gelijk blijft aan de testhoeveelheid, deze is dus vanaf het moment van vuren gelijk aan het verschil tussen de instroom en de uitstroom. Dit betekent dat deze pijl afwijkt van al eerder gedefinieerde pijlen in de zin dat hij twee functies heeft. Allereerst test hij de plaats en vervolgens past hij de stroomsnelheid van de transitie aan [1]. Dit is te zien in figuur 6, waarbij de pijl van $\mathcal{T}_3$ naar $\mathcal{T}_{5a}$ een overstroom is en het overstroomde water terecht komt in plaats $\mathcal{P}_{3a}$. Deze plaats is in werkelijkheid niet fysiek aanwezig, maar door naar de inhoud hiervan te kijken, kan daardoor bepaald worden hoe vaak en hoeveel de buffer overstroomt.

Wanneer een bestaand programma wordt gebruikt om een Petrinet te modelleren is het niet altijd

[1] Een dergelijke constructie waarbij een pijl een ander soort functie vervult is te zien bij de flush-out pijl. Deze leegt de inhoud van een plaats geheel, wanneer de bijbehorende transitie vuurt [9].



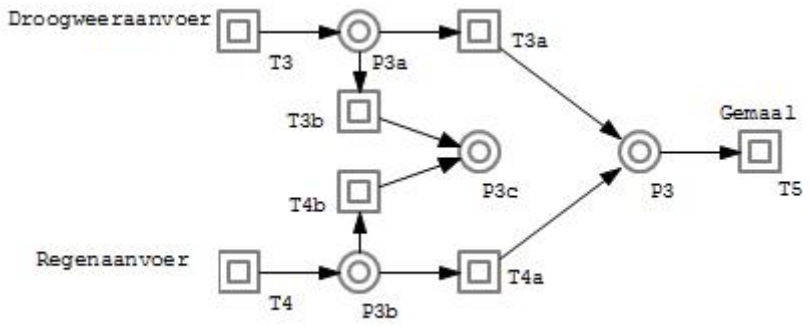
Figuur 6: Uitbreiding overstroompijl. De pijl van $\mathcal{T}_3$ naar $\mathcal{T}_{5a}$ is een overstroompijl, die pas actief is wanneer de buffer vol is.

mogelijk om het programma uit te breiden met bijvoorbeeld een continue testpijl [2]. Om deze reden is bovenstaand principe ook op een andere manier gemodelleerd, waarbij slechts gebruikt gemaakt wordt van bestaande elementen.

De wens is om het teveel aan water dat naar de bufferplaats stroomt op te vangen in een extra overstroomplaats. Hiervoor kan in eerste instantie gebruik gemaak worden van de standaard functionaliteit dat de instroom wordt verminderd als een plaats zijn bovengrens bereikt. Dit betekent dat het overschot aan water wordt verplaatst naar de plaats vóór deze verminderde transitie. Dit overschot aan water moet vervolgens naar een opvangplaats stromen door een continue transitie. Echter, door deze transitie zou alleen water moeten stromen in het geval dat de standaard transitie de invoer niet aan kan. Hiervoor wordt de prioriteitsconstructie gebruikt. Een transitie met hogere prioriteit transporteert, indien genoeg capaciteit, alle vloeistof. Bij een te kleine capaciteit, bijvoorbeeld bij vermindering door een volle buffer, zal een transitie met lagere prioriteit ook water verplaatsen.

Een Petrinetmodel van deze constructie is te zien in figuur 7. Allereerst wordt de droogweeraanvoer naar de buffer opgedeeld in twee verschillende transities ($\mathcal{T}_3$ en $\mathcal{T}_{3a}$) met eenzelfde stroomsnelheid. Hiertussen zit een plaats $\mathcal{P}_{3a}$ met een bovengrens van nul. Vanuit deze plaats is er nog een extra transitie die de overstroomhoeveelheid weergeeft ($\mathcal{T}_{3b}$).
Door de pijl naar transitie $\mathcal{T}_{3a}$ prioriteit te geven over de pijl naar transitie $\mathcal{T}_{3b}$, zal deze laatste aanvankelijk niet actief zijn, aangezien de stroomsnelheden van $\mathcal{T}_3$ en $\mathcal{T}_{3a}$ gelijk zijn. Al het water stroomt dan naar de buffer $\mathcal{P}_3$. Wanneer de buffer echter vol is en er in de buffer geldt dat de instroom groter is dan de uitstroom, wordt de stroomsnelheid van $\mathcal{T}_{3a}$ omlaag gebracht. Hierdoor is er in $\mathcal{P}_{3a}$ een grotere instroom dan uitstroom. Aangezien er in $\mathcal{P}_{3a}$ niks opgeslagen kan worden, zal op dit moment de extra instroom gecompenseerd worden door $\mathcal{T}_{3b}$ te activeren en hierdoor het extra water weg te laten stromen naar plaats $\mathcal{P}_{3c}$. Hierin is dan precies te zien hoeveel water er teveel voor de buffer was. Het water dat teveel is op het moment dat het regent komt door eenzelfde constructie in deze zelfde plaats terecht, waardoor hier de totale hoeveelheid overstroomd water wordt opgevangen.

[2] Dit geldt ook voor het analyse programma FST dat in sectie 5.1 gebruikt zal worden. Een dergelijke pijl is niet beschikbaar in de gebruikte versie van de tool.



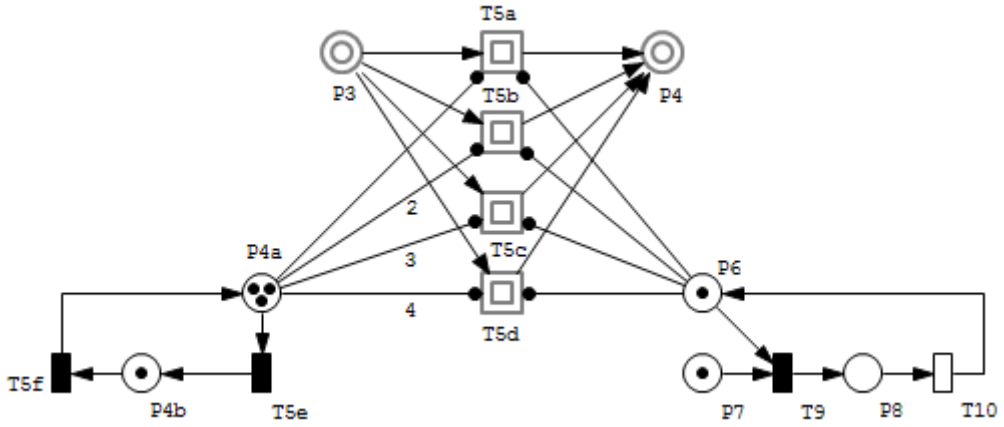
Figuur 7: Uitbreiding opslag overstroomwater. De pijl naar $\mathcal{T}_{3a}$ heeft prioriteit over de pijl naar $\mathcal{T}_{3b}$ en de pijl naar $\mathcal{T}_{4a}$ heeft prioriteit over de pijl naar $\mathcal{T}_{4b}$. Hierdoor komt al het teveel aan water in $\mathcal{P}_{3c}$ terecht.

4.2.2 Meerdere gemalen

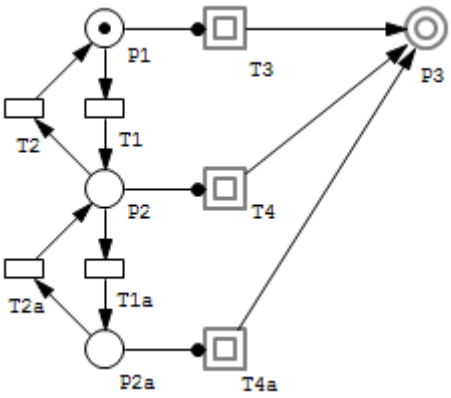
De aanvoer van de waterzuivering wordt geregeld door vier verschillende gemalen, die allemaal een deel van het water naar binnen pompen. Wanneer bijvoorbeeld de stroom uitvalt staan deze allemaal stil. Het kan echter ook zo zijn dat er slechts één gemaal kapot gaat. Dit resulteert in een kleinere capaciteit van de waterzuivering. Om dit te representeren is in figuur 8 de invoer weergegeven als vier verschillende transities,$\mathcal{T}_{5a}$, $\mathcal{T}_{5b}$, $\mathcal{T}_{5c}$ en $\mathcal{T}_{5d}$. Deze transities testen plaats $\mathcal{P}_{4a}$ op het aantal aanwezige tokens, dat de werking van het aantal gemalen geeft. De gewichten ϕ_w^A van de pijlen hiertussen zijn daarom respectievelijk één, twee, drie en vier. Zo is gemaal $\mathcal{T}_{5a}$ altijd werkend als er minstens één token is en gemaal $\mathcal{T}_{5d}$ alleen als er minstens vier tokens zijn. Verder werken alle gemalen alleen als er een token in plaats $\mathcal{P}_6$ is, wat aangeeft of er geen sprake is van bijvoorbeeld stroomuitval, waardoor alle gemalen buiten werking zijn. Aangezien is aangenomen dat alle gemalen soortgelijk zijn, duurt het voor iedere pomp even lang voordat hij stuk gaat en is ook de reparatietijd van de gemalen gelijk gekozen. Transitie $\mathcal{T}_{5e}$ geeft de tijd voordat een pomp kapot gaat en $\mathcal{T}_{5f}$ geeft vervolgens de reparatieduur aan. De gemalen gaan dus beurtelings kapot en er kan slechts één pomp tegelijk gerepareerd worden.

4.2.3 Verschil in regenaanvoer

Aangezien er subtielere verschillen in de aanvoer zitten dan alleen droog en regen is een uitbreiding nodig om de aanvoer weer te geven als meerdere transities. Deze uitbreiding is te zien in figuur 9. Hierbij moet gelden dat er niet ineens aanvoer horend bij stortregen kan zijn ($\mathcal{T}_{4a}$) als er daarvoor sprake was van droogweer aanvoer ($\mathcal{T}_3$). Het token kan daarom vanuit $\mathcal{P}_1$ alleen rechtstreeks naar $\mathcal{P}_2$, wat de overgang van droogweer aanvoer naar gemiddelde aanvoer aangeeft en niet direct naar $\mathcal{P}_{2a}$, wat de overgang naar stortregen aanvoer zou geven.
Verder is het wenselijk om de transities $\mathcal{T}_1$, $\mathcal{T}_2$, $\mathcal{T}_{1a}$ en $\mathcal{T}_{2a}$ die de overgangen tussen het weer aangeven, stochastisch te maken. In werkelijkheid is de aanvoer namelijk ook stochastisch in plaats van deterministisch.



Figuur 8: Uitbreiding met meerdere gemalen. In bovenstaande toestand is er één gemaal stuk en werken daardoor alleen transities $\mathcal{T}_{5a}$, $\mathcal{T}_{5b}$ en $\mathcal{T}_{5c}$, wat 75% van de totale invoer geeft.

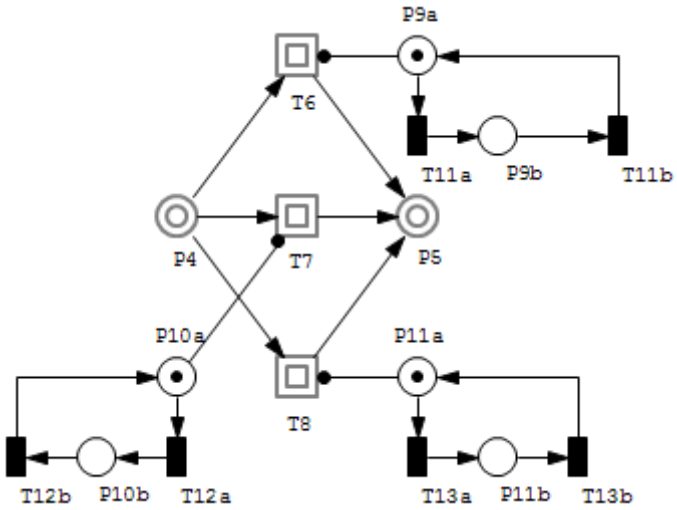


Figuur 9: Uitbreiding met verschil in regenaanvoer. Wanneer het token zich in plaats $\mathcal{P}_1$ bevindt is transitie $\mathcal{T}_3$ actief en is er sprake van droogweer aanvoer. Als het token in $\mathcal{P}_2$ is er sprake van gemiddelde aanvoer gegeven door $\mathcal{T}_4$ en bij $\mathcal{P}_{2a}$ van stortregen aanvoer $\mathcal{T}_{4a}$.

4.2.4 Stillegging afzonderlijke zuiveringsstraten

Wanneer er een extra giftige lading komt, kan het gunstig zijn een deel van de zuiveringsstraten tijdelijk af te sluiten, zodat deze niet belast worden. Ook kan het zo zijn dat er een enkel gemaal bij een van de zuiveringsstraten kapot gaat. Daarom is een uitbreiding nodig, waarbij iedere zuiveringsstraat apart kapot kan gaan of uitgezet kan worden. Allereerst betekent dit dat het zuiveringsproces gescheiden moet worden van de zandvang, die slechts eenmaal aanwezig is. De zandvang wordt nu weergegeven als een continue plaats ($\mathcal{P}_4$) met een bovengrens van nul. Dit betekent dat alles wat erin gaat er ook weer meteen uit moet kunnen, aangezien het een bassin vol water is.
Na de zandvang wordt het proces vervolgens opgedeeld in drie verschillende straten met verschil-

lende capaciteiten, waarbij elke straat individueel aan of uit kan staan. Aangezien elke straat een eigen gemaal heeft die het debiet regelt, worden deze gemalen weergegeven als continue transities $\mathcal{T}_6$, $\mathcal{T}_7$ en $\mathcal{T}_8$. Om de werking van ieder gemaal te modelleren zijn bij iedere straat twee discrete plaatsen toegevoegd. Hiertussen beweegt zich één token, dat weergeeft wat de toestand van het betreffende gemaal is, aan of uit. In figuur 10 is een toestand te zien waarbij alle gemalen in werking zijn. Wanneer het token in $\mathcal{P}_{9a}$ verspringt naar $\mathcal{P}_{9b}$, betekent dit dat de eerste straat ($\mathcal{T}_6$) niet meer in werking is. De totale capaciteit van de zuivering is op dan ook kleiner. Bij deze pomp geeft vervolgens $\mathcal{T}_{11b}$ de reparatietijd voordat de pomp weer in werking is. Plaats $\mathcal{P}_5$ geeft vervolgens weer de totale hoeveelheid gezuiverd water, dat gelijk is aan het water van de drie straten samen.



Figuur 10: Uitbreiding met stillegging afzonderlijke zuiveringsstraten. In deze toestand geven de tokens in $\mathcal{P}_{9a}$, $\mathcal{P}_{10a}$ en $\mathcal{P}_{11a}$ aan, dat alle gemalen in werking zijn.

5 Implementatie

Nu een model is opgesteld, is een simulatie en analyse omgeving nodig om hierover uitspraken te doen. In dit hoofdstuk worden twee omgevingen geïntroduceerd waarmee het gedrag van een Petrinet bekeken kan worden. Allereerst komt een bestaand programma aan bod, waarmee Petrinetten geanalyseerd kunnen worden. Aangezien hierin niet alle uitbreidingen op het basismodel opgenomen kunnen worden, is daarnaast een eigen simulatieprogramma ontwikkeld met uitgebreidere functionaliteiten. Het bestaande programma kan vervolgens gebruikt worden om de werking hiervan te controleren. Beide programma's hebben hun eigen voor- en nadelen, daarom zal aan het eind van dit hoofdstuk een vergelijking worden gemaakt tussen beide omgevingen. Hierin wordt afgewogen in welke situatie welk programma het best bruikbaar is.

5.1 Analyse met Fluid Survival Tool

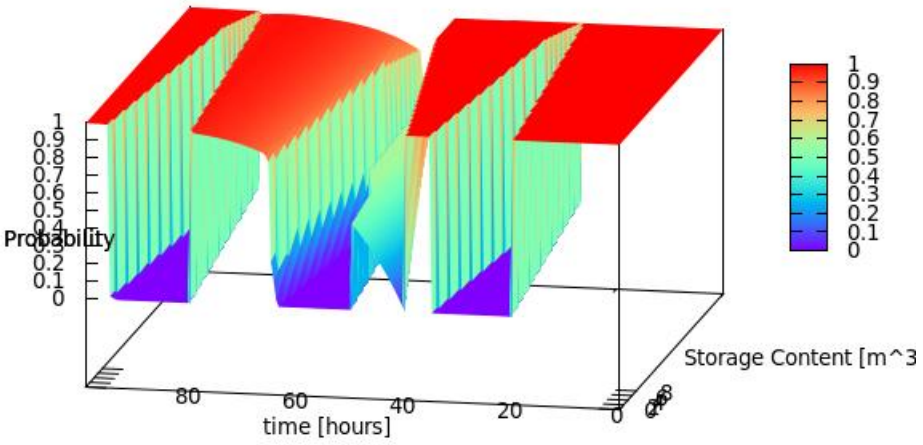
Om het basismodel te analyseren wordt er gebruik gemaakt van de Fluid Survival Tool (FST) [8], [1]. Bij gebruik van FST wordt het model handmatig ingevoerd. Vervolgens kan een plaats uit het model worden gekozen waarvan de inhoud geanalyseerd wordt.

Input Allereerst wordt er een lijst met continue en discrete plaatsen gedefinieerd. Verder wordt er voor discrete plaatsen het aantal tokens in de begintoestand gegeven en voor continue plaatsen de beginhoeveelheid en de bovengrens.
Er kan gebruik worden gemaakt van vier verschillende soorten transities, continue, deterministische, onmiddellijke en stochastische transities. Ook deze worden in een lijst gedefinieerd, met de restrictie dat er slechts één stochastische transitie in het model kan zijn, die ook slechts eenmaal kan vuren. Voor een deterministische transitie moet de tijd voordat hij vuurt worden ingevoerd, voor een continue transitie de stroomsnelheid en voor een stochastische transitie de kansverdeling. Verder moet van alle transities het gewicht en de prioriteit gegeven worden.
Als laatste zijn er zes verschillende pijlen, discrete input, discrete output, continue input, continue output, test pijlen en stop pijlen. Van de pijlen wordt gegeven waar ze vandaan komen, waar ze naartoe gaan en hun gewicht, aandeel en prioriteit.

Output Wanneer het Petrinet op deze manier geschematiseerd is, kan een plaats $\mathcal{P}$ met inhoud m worden gekozen om te analyseren. Op deze manier kan gekeken worden wat er gebeurt met de inhoud van deze plaats wanneer de tijd loopt. Hiervoor wordt $P(m \leq c)$ berekend voor iedere inhoud c, waarvoor geldt $c \leq \phi_b^{\mathcal{P}}$, met $\phi_b^{\mathcal{P}}$ de bovengrens van plaats $\mathcal{P}$. De tool doorloopt met een gekozen stapgrootte c van een bepaalde ondergrens tot een bepaalde bovengrens, bijvoorbeeld voor de buffer van nul tot de maximum van de buffer. Deze kansen geven de verdelingsfunctie van de inhoud. Deze verdelingsfunctie wordt berekend voor elk tijdstap, met een totale tijdsduur van 100 uur. Hierbij is ook de stapgrootte van de tijdstap te kiezen. Uiteindelijk geeft dit een 3D-grafiek, waarbij voor alle mogelijke inhouden c en tijdstippen t de kans wordt gegeven dat de inhoud van de betreffende plaats op dat tijdstip kleiner of gelijk aan c is. Het is ook mogelijk de exacte kansen als output in een tabel te krijgen in plaats van grafisch weergegeven.

Een voorbeeld van een 3D-grafiek die de FST zou kunnen maken is te zien in figuur 11. Het geeft de kans dat de inhoud van de buffer (storage content) na een bepaalde tijd kleiner of gelijk is aan een bepaalde waarde. De situatie waarvoor deze grafiek gemaakt is bestaat uit droge periodes van 20 uur en regenperiodes van 10 uur, zoals te zien is tot tijdstip 20 de kans 1 dat de inhoud kleiner is dan elke waarde. Dit betekent dat de buffer leeg is.Op tijdstip 20 begint het eerste water de buffer in te stromen en even na tijdstip 30 is de inhoud van de buffer weer verwerkt en

is hij opnieuw leeg. Op tijdstip 40 valt de stroom uit en kan de installatie geen water verwerken waardoor de buffer volloopt. De reparatietijd is exponentieel verdeeld waardoor de onzekerheid over de bufferinhoud onstaat na tijdstip 40. Na tijdstip 80 is de zekerheid weer teruggekeerd aangezien de buffer dan zeker leeg moet zijn.



Figuur 11: Verdelingsfunctie van de waterhoogte als functie van de tijd bij analyse met de Fluid Survival Tool. Op de verticale as zijn de kansen gegeven dat de bufferinhoud kleiner of gelijk is aan een inhoud en op de horizontale assen de inhoud en de tijd. In dit figuur is bijvoorbeeld af te lezen dat op tijdstip 10 er een kans is van 1 dat hij kleiner of gelijk aan iedere inhoud is. Dit betekent dat de betreffende plaats op dit tijdstip met 100% zekerheid leeg is.

5.1.1 Voordelen

- De tool geeft een exacte kansverdeling voor de inhoud van een plaats.
- De tool berekent binnen enkele seconden de kansverdeling voor een plaats, ongeacht de verdeling van de stochastische transitie.
- Het model is gemakkelijk in te voeren, hiervoor hoeft slechts een lijst met alle plaatsen, transities, en pijlen gegeven te worden. Ook is het duidelijk op welke manier de eigenschappen van de plaatsen en transities gedefinieerd kunnen worden.
- Het model is gemakkelijk aan te passen. Wanneer er bijvoorbeeld een aantal extra plaatsen of transities toegevoegd moeten worden kan dit eenvoudig gedaan worden zonder de rest van het model aan te hoeven passen. Ook eigenschappen zoals de bovengrens van een plaats of de stroomsnelheid van een transitie kunnen heel eenvoudig aangepast worden. Hierdoor kunnen er gemakkelijk verschillende situaties geanalyseerd worden.

5.1.2 Nadelen

- Bij gebruik van FST kan slechts één transitie een stochastische vuurtijd hebben, die ook slechts eenmaal kan plaatsvinden (vuren). Hierdoor is een model met meerdere kansverdelingen niet te analyseren met FST zonder het eerst te versimpelen.
- Er is nog geen grafische weergave van het Petrinet. Hierdoor is het lastig te zien welk Petrinet is gebruikt wanneer het door iemand anders is ingevoerd.
- Tijdens de opdracht was het voor ons niet mogelijk om uitbreidingen in het model, zoals een continue testpijl, toe te voegen. Ook was de tool enkel te gebruiken in Linux[3].

5.2 Simulatie met C++

Aangezien het petrinet uit de modelformulering in het algemeen te uitgebreid is om te analyseren met behulp van de Fluid Survival Tool, is gekozen een simulatie programma te schrijven met behulp van de programmeertaal C++. Op deze manier kan namelijk gebruik gemaakt worden van meerdere stochastische transities, die ook allemaal meerdere malen kunnen vuren.

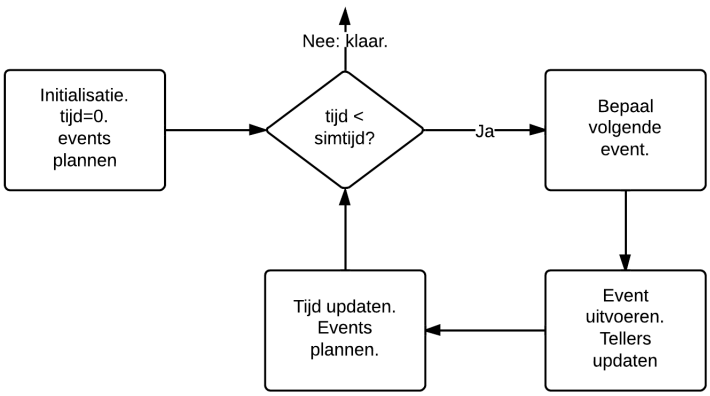
Het simuleren van een enkel proces met behulp van Discrete Event Simulatie (DES) kost voor het gekozen model van de waterzuivering weinig rekentijd. Tussen de momenten waarop er iets in het systeem verandert zijn de in- en doorstroomsnelheden constant. De momenten waarop het systeem verandert zijn bijvoorbeeld stroomuitval of verandering in aanvoerhoeveelheid. Omdat het debiet tussen de gebeurtenissen constant zijn, zijn de inhouden van de continue plaatsen op alle tijdsintervallen tussen de veranderingen lineair. Hierdoor is het eenvoudig om op elk willekeurig moment inhouden van de verschillende continue plaatsen te bepalen.

Het proces maakt op deze manier een aantal verschillende veranderingen mee, zogenaamde events. In de tijd tussen deze events is het procesverloop bekend en kan daardoor de inhoud van de buffer op verschillende tijdstippen benaderd worden. De verschillende events die kunnen optreden zijn verandering van de invoer, verandering van de uitvoer en overstromen of leeglopen van een plaats. In het programma wordt allereerst bepaald welke events zich kunnen voordoen en wanneer deze plaatsvinden. Zodra een bepaald event heeft plaats gevonden worden alle events, die door dit event mogelijk worden, ingepland. Na het plaatsvinden van ieder volgend event wordt bepaald wat het eerstvolgende event is. In een flowchart ziet het programma eruit als in figuur 12. Op deze manier kan het programma op ieder tijdstip bepalen wat de inhoud is van elke willekeurige plaats.

Stochastische events Voor het simuleren van stochastische events wordt een kansverdeling gebruikt om de eventtijd te bepalen. Voor het benaderen van deze kansverdelingen wordt gebruik gemaakt van willekeurig gegenereerde getallen, getrokken uit een vooraf aangenomen kansverdeling. Door een groot aantal runs te simuleren en de resultaten uit te middelen ontstaat een benadering voor de toestanden, met bijbehorende kans, waar het netwerk zich in kan bevinden met een dergelijke willekeurige event. De nauwkeurigheid en betrouwbaarheid van deze methode wordt later in dit hoofdstuk besproken.

Het bepalen van stochastische events gaat als volgt. Het programma genereert willekeurige getallen tussen 0 en 1. Voor een uniforme verdeling over het interval (a, b), wordt een willekeurig

[3]Inmiddels is de tool ook beschikbaar op andere platformen.



Figuur 12: Flowchart van het `c++` programma.

getal direct gelinkt aan een getal uit dat interval door het te vermenigvuldigden met het verschil tussen a en b en daarna a erbij op te tellen. Voor het construeren van de exponentiële verdeling met een willekeurig getal x, uniform verdeeld tussen 0 en 1, wordt vergelijking 1 gebruikt om de tijd T_{exp} wanneer het event plaatsvindt te bepalen, waarbij λ de parameter van de exponentiële verdeling is. Deze verdelingen worden vervolgens gebruikt om tijdstip van het plaatsvinden van de stochastische events te simuleren.

$$T_{exp} = \frac{-1}{\lambda} \log (x + 1) \tag{1}$$

Het programma benadert voor elke tijdstap de kans dat de inhoud van een plaats kleiner gelijk is dan een bepaalde inhoudsgrens. De kansverdeling die te vergelijken is met de output van de FST. Met een kleine aanpassing zou ook benaderd kunnen worden of de bufferinhoud op een tijdstap met een bepaalde kans in een bepaald interval voor de bufferinhoud ligt. De doelstelling is echter om te bekijken of de plaats overstroomt en daardoor is de kans dat de inhoud kleiner gelijk is aan een waarde interessant om te weten, is de kans dat de inhoud onder zijn maximumwaarde wel gelijk aan 1.

Met deze aanpak kunnen allerlei verschillende situaties worden onderzocht, met een stochastische verdeling voor bijvoorbeeld de reparatietijd kan beschouwd worden wat voor effect een dergelijke onzekerheid heeft. Door te variëren met verschillende verwachtingswaarden voor de reparatietijd, met verder realistische parameters, kan onderzocht worden of de kans, of hoeveel de kans, op overstroming verminderd zou kunnen worden door te investeren in een kortere reparatietijd.

Rekentijd De rekentijd die het simulatieprogramma nodig heeft lijkt erg lineair ten opzichte van het aantal runs dat gesimuleerd wordt, zie tabel 1. Dit is vooral omdat het simuleren van 1 run de grootste taak van het programma is en het programma daarom heen vrij weinig computaties uitvoert. Een simulatie met $10 \times n$ runs vergt daardoor ongeveer 10 maal zoveel

rekentijd als een simulatie met n runs, in het geval n groot genoeg. De computaties van een enkele run zijn niet erg groot ten opzicht van de rest van de computaties.

Aantal runs	Simulatietijd in milliseconden
1000	312
100000	29530
1000000	308625

Tabel 1: Rekentijden voor simulatie van het netwerk met 3 stochastische eventtijden

5.2.1 Voordelen

- Er kunnen meerdere stochastische transities in het Petrinet gesimuleerd worden, die ook meerdere malen kunnen vuren.
- Uitbreidingen van het Petrinet, zoals een continue testarc, kunnen relatief eenvoudig worden toegevoegd.
- Verschillende trekkingen kunnen onafhankelijk van elkaar gesimuleerd worden, waardoor één simulatie op meerdere processoren parallel gesimuleerd kan worden. Een simulatie van een complex netwerk waarbij een groot aantal runs nodig is kan hierdoor uitgevoerd worden, wanneer voldoende rekenkracht ter beschikking is.

5.2.2 Nadelen

- De simulatieaanpak geeft een benadering van de kansverdeling. Bij sommige verdelingen zijn veel runs nodig voor een acceptabele benadering.
- Het programma is op dit moment modelspecifiek. Wanneer een ander model bekeken wordt moet dit opnieuw geïmplementeerd worden en is programmeerkennis nodig.
- De hoeveelheid rekentijd wordt groter naarmate de varianties van de stochastische transities toenemen, aangezien er dan meer runs nodig zijn. Ook bij een netwerk met heel veel events is mogelijk veel rekentijd nodig.
- Er is geen grafische weergave van het Petrinet. Hierdoor is programmeerkennis nodig om te zien welk Petrinet is gemodelleerd.

5.2.3 Betrouwbaarheid van de simulatie

Het programma maakt gebruikt van willekeurig genereerde getallen om een kansverdeling te benaderen. Om uitspraken te doen over hoe goed deze benadering is en hoe accuraat het programma dus is, kan gebruik gemaakt worden van een betrouwbaarheidsinterval. De vraag is namelijk hoeveel runs gedraaid moeten worden voor een accurate benadering. Ook is het interessant hoeveel accurater deze benadering wordt wanneer er meer runs gesimuleerd worden en of deze verbetering significant is.

Wanneer FST een plaats analyseert, genereert het voor ieder tijdstip t en iedere grenswaarde c,

de kans dat de inhoud m van die plaats op dat tijdstip onder deze grenswaarde ligt. Wanneer de toestand $c \leq m$ wordt gedefinieerd als een succes, is deze kansverdeling binomiaal verdeeld. Dit betekent dat de inhoud met kans p onder of op de grenswaarde ligt en met kans $1-p$ erboven.

Het benaderen van de kans met de simulatie gaat als volgt. Een enkele run bepaalt voor elk tijdstip en elke grenswaarde, of de inhoud op dat moment onder of boven de grenswaarde ligt. Hierbij wordt inhoud onder of op de grenswaarde beschouwd als een succes. Bij een simulatie wordt in een aantal runs n het aantal successen X registreerd. De schatter $\hat{p}$ benadert dan de kans op succes p van de kansverdeling:

$$\hat{p} = \frac{X}{n}. \tag{2}$$

Wanneer geldt dat het aantal runs n groot genoeg is en dat de succeskans p niet te klein of te groot is, kan de binomiale verdeling benaderd worden door een normale verdeling. Aangezien geldt dat $E[X] = np$ en $var(X) = np(1-p)$, geldt bij n groot genoeg $X \sim N(\ np\ ,\ np(1-p)\ )$. Voor de constructie van een betrouwbaarheidsinterval met betrouwbaarheid γ kan er dan een c gevonden worden waarvoor geldt

$$\gamma\ = P(|Z| \leq c)\ \approx P(\frac{|X-np|}{\sqrt{np(1-p)}} \leq c),$$

met Z standaard normaal verdeeld. Omdat het een benadering is en er bij voldoende runs n geldt dat X en $E[X]$ relatief weinig verschillen, kan dan $np(1-p)$ benaderd worden met $X(1-X/n)$. [2] Hieruit volgt voor benadering met betrouwbaarheid γ

$$|X-np| \leq c\ \sqrt{X(1-\frac{X}{n})},$$

$$|p-\frac{X}{n}| \leq \frac{c\ \sqrt{X(1-\frac{X}{n})}}{n}.$$

Het betrouwbaarheidsinterval voor $\hat{p}$ ziet er dan als volgt uit

$$\left(\frac{X}{n} - \frac{c\ \sqrt{X(1-\frac{X}{n})}}{n},\ \ \frac{X}{n} + \frac{c\ \sqrt{X(1-\frac{X}{n})}}{n}\right).$$

Wat door substitutie van formule 2 ook geschreven kan worden als

$$\left(\hat{p} - c\ \sqrt{\frac{\hat{p}(1-\hat{p})}{n}},\ \ \hat{p} + c\ \sqrt{\frac{\hat{p}(1-\hat{p})}{n}}\right). \tag{3}$$

Voor het programma is een betrouwbaarheid van 95% gewenst. Wanneer de exacte uitkomsten binnen dit betrouwbaarheidsinterval vallen, is het model accuraat genoeg. Dit betekent dat we bij de verificatie c kiezen zodanig dat $0.95\ = P(|Z| \leq c)$.

5.3 Vergelijking tussen simulatie en FST

Het grootste voordeel van het C++ programma ten opzichte van FST is de mogelijkheid tot het toevoegen van meerdere verschillende stochastische transities met ook meerdere vuurmomenten. Waar FST slechts één stochastisch moment van vuren kan implementeren, kunnen aan het C-programma meerdere worden toegevoegd. Hierbij moet wel worden opgemerkt dat er dan waarschijnlijk (veel) meer runs nodig zijn om de gewenste nauwkeurigheid te bereiken.
Een nadeel van het simulatie programma is echter dat het alleen geschikt is voor dit specifieke model. Een ander Petrinet kan in principe op dezelfde wijze gesimuleerd worden, maar het vereist kennis van programmeren om de specifieke events en transities voor dat netwerk te verwerken. Dit in tegenstelling tot FST, waar gemakkelijk een nieuw model in te voeren is. Bovendien wordt er geen exacte analyse gedaan, waardoor het een benadering geeft van het gedrag van het Petrinet. Ook is niet van te voren duidelijk hoeveel runs er gemaakt moeten worden om de kansverdeling met gewenste nauwkeurigheid te benaderen. Dit kan veel tijd kosten, terwijl FST altijd binnen enkele seconden het netwerk heeft geanalyseerd. Dit betekent dat FST het best gebruikt kan worden wanneer er exacte uitspraken gedaan moeten worden over verschillende simpele modellen. Wanneer het model meerdere stochastische transities bevat, kan het best gebruik worden gemaakt van een simulatie programma.

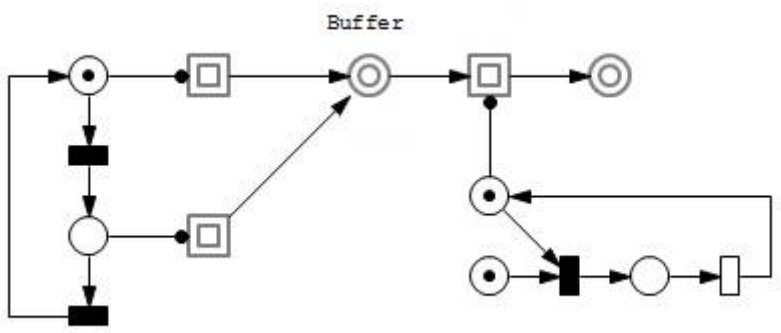
Als deze manier van het modelleren en simuleren van een Petrinet erg praktisch en betrouwbaar blijkt, is het schrijven van een programma waar een willekeurig petrinet als input gegeven kan worden een goede uitbreiding. De manier van simuleren is voor het modelleren van hele grote netwerken namelijk interessant. Uitgebreide Petrinetten hebben namelijk een erg grote toestandsruimte, die vaak te groot is om met een analytische tool te bepalen. Door middel van een groot aantal maal simuleren van het netwerk kunnen de bereikbare toestanden worden bepaald. Hoe vaak gesimuleerd moet worden hangt dan af van de grootte van de toestandsruimte. Bij een grotere ruimte is een groter aantal trekkingen nodig om een effectieve benadering te krijgen. Voor een heel uitgebreid netwerk zouden dit heel veel runs kunnen zijn, maar aangezien dit parallel gesimuleerd kan worden, kan de rekentijd verminderd worden door hier meerdere processoren voor te gebruiken.

6 Validatie en verificatie

Voordat het simulatieprogramma in gebruik kan worden genomen, moet eerst de werking gecontroleerd worden. Hiervoor is validatie en verificatie van het programma nodig. Aangezien FST alleen Petrinetten met maximaal één stochastische transitie kan analyseren, hebben we deze uitgevoerd aan de hand van het basismodel wat in sectie 4.1 is opgesteld.

6.1 Modelparameters voor basismodel

Figuur 13 beschrijft een installatie waarbij het gemaal een capaciteit heeft van 12.000 m^3/h. De aanvoer van het water is bij droog weer 4.000 m^3/h en bij regen 20.000 m^3/h. Het weer is deterministisch, waarbij het afwisselend 20 uur droog is en 10 uur regent. De inhoud van de buffer is maximaal 150.000 m^3. Verder valt na 40 uur de elektriciteit uit, waardoor het gemaal niet meer werkt en de installatie geen water meer kan verwerken. Om de werking van de installatie in verschillende situaties te vergelijken is de reparatietijd bij stroomuitval gevarieerd. De reparatietijd is hierbij deterministisch, uniform of exponentieel verdeeld.

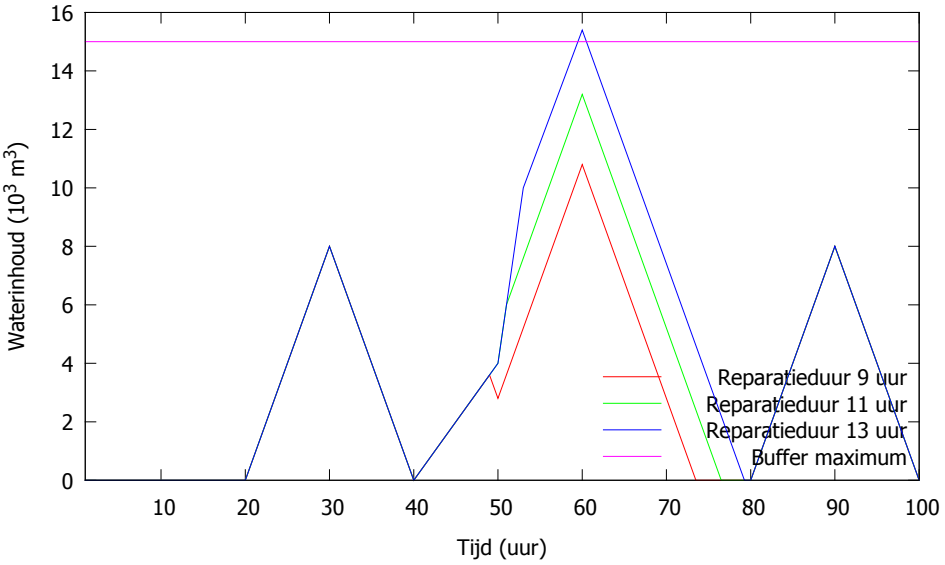


Figuur 13: Petrinet van de installatie die gebruikt is voor de validatie. De plaats die zal worden bekeken is aangegeven als buffer.

De plaats die geanalyseerd wordt is in dit geval de buffer. De hoeveelheid water in deze continue plaats wordt gegeven door stochast X. Aangezien deze verdeling met de tijd verandert kan deze worden beschreven met een continu stochastisch proces $\{X_t\}_{t\in\mathbb{R}}$. De validatie van de simulatie gebeurt door de verdelingsfunctie met de simulatie te benaderen en deze vervolgens met de verdelingsfunctie zoals die door de tool is bepaald te vergelijken.

6.2 Verificatie

Allereerst wordt gegeken of het model een goede weerspiegeling van de werkelijkheid geeft. Hiervoor is een bekende, deterministische situatie gebruikt, zodat handmatig geverifieerd kan worden of de resultaten correct zijn. Hiervoor zijn reparatietijden van respectievelijk 9, 11 en 13 uur bekeken. De stroom valt uit op tijdstip 40. Dit levert de grafiek op die te zien is in figuur 14.

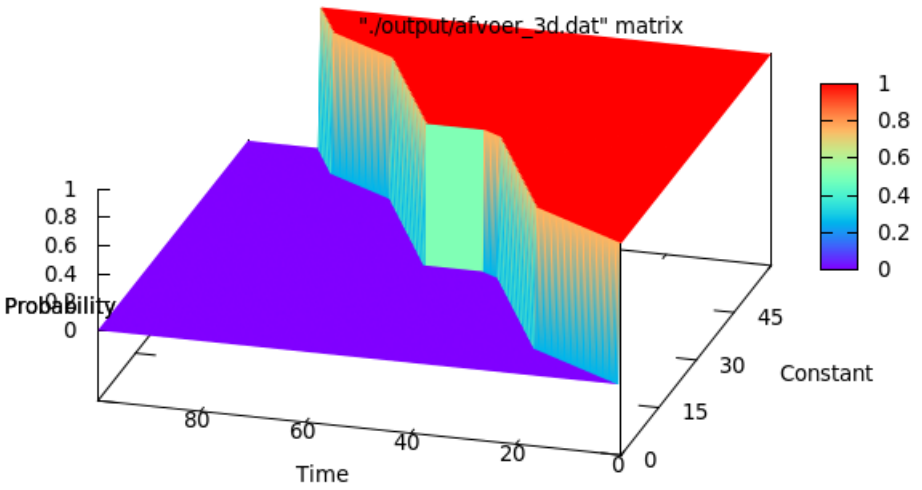


Figuur 14: De inhoud van de buffer voor drie verschillende reparatietijden. Er is te zien dat op tijdstip 20 de regen begint en het op tijdstip 30 weer droog wordt. De pomp gaat vervolgens kapot op tijdstip 40 en wordt respectievelijk op tijdstip 49, 51 en 53 gerepareerd. Alleen bij een reparatieduur van 13 uur vindt er een overstroming plaats.

Er is te zien dat de buffer sneller vol loopt wanneer het gemaal kapot is en de reparatietijd langer is gekozen. Daarnaast valt op dat de buffer alleen overstroomt bij een reparatieduur van 13 uur. Verder zijn met een handsimulatie alle events gecontroleerd, die in de grafiek als omslagpunten te zien zijn. Events zijn in dit geval de start van regen op tijdstip 20, 50 en 80. De beëindiging van regen op tijdstip 30, 60 en 90. Het uitvallen van het gemaal op tijdstip 40 en de het eind van de reparatie van het gemaal op respectievelijk tijdstip 49, 51 en 53. Op al deze tijdstippen is direct gecontroleerd of de bijbehorende inhoud van de buffer correct is. De eerste 20 uur is het droog waardoor al het water dat binnenkomt ook verwerkt kan worden, hierdoor heeft de buffer op dit moment nog geen inhoud. Op $t = 30$ heeft het net 20 uur geregend, waardoor de instroom gelijk is aan 20.000 m^3/h terwijl er slechts $12.000m^3/h$ water verwerkt kan worden. Dit betekent dat er $(20.000 - 12.000) * 10 = 8.000m^3$ water in de buffer aanwezig is voordat de buffer weer begint leeg te lopen. Aangezien er vanaf t=30 een droge periode van 10 uur intreedt. Dit komt overeen met figuur 14. Ook is te zien dat op $t = 40$ de storing plaatsvindt, terwijl het droog is, waarna op $t = 50$ het weer begint te regenen. Ook de gegevens van de inhoud op deze tijdstippen kloppen met de handsimulatie.

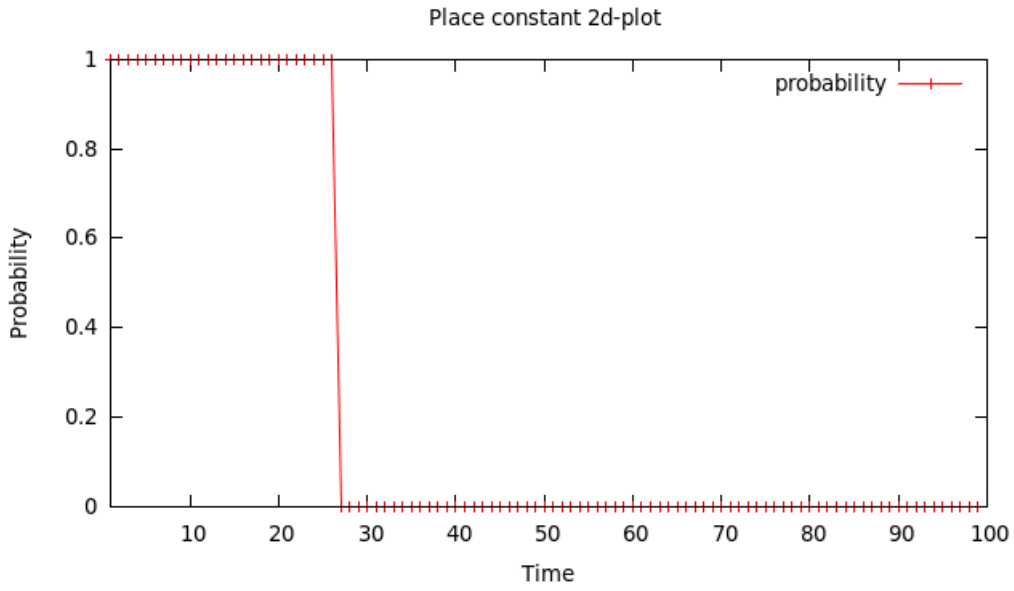
6.3 Validatie uitbreiding overstroompijl

In sectie 4.2.1 wordt gesproken over de uitbreiding van het model om de buffer te laten overstromen wanneer deze vol raakt. Aangezien een overstroompijl niet aanwezig is in de gereedschapskist van FST is een ander model opgesteld met dezelfde werking. Om te controleren of deze uitbreiding ook daadwerkelijk werkt in FST is het model uitgebreid met de nodige plaatsen en transities met bijhorende prioriteiten. Hiermee is de deterministische situatie zoals in sectie 6.1 geanalyseerd. Het gaat zoals te zien is in de figuur 15 vanaf tijdstip 20 regenen, wat betekent dat er dan 20.000 m^3/h binnenkomt, maar slechts 12.000 m^3/h door de installatie wordt verwerkt. Dit geeft 8.000 m^3/h die niet verwerkt kan worden. Bij een buffer van 50.000 m^3/h betekent dit dat na zes uur en een kwartier de buffer begint te overstromen.



Figuur 15: Hoeveelheid gezuiverd water in het deterministische geval met FST. Aangezien het een deterministische situatie is, is er alleen een kans van 0 of 1 en geeft de spronglijn de hoeveelheid water aan. Er is te zien dat tussen tijdstip 0 en 20 een constante hoeveelheid water binnenkomt van 4.000 m^3/h, die correspondeert met de hoeveelheid water bij droogweeraanvoer. Vervolgens komt er tussen tijdstip 20 en 30 een hoeveelheid van 12.000 m^3/h binnen, de maximale capaciteit. Hierna is er een periode geen gezuiverd water aangezien het gemaal niet werkt.

Dit betekent dat vanaf tijdstip 26.25 de overstroomcontainer als eerste water binnen zou moeten krijgen. Een analyse met tijdstap 1 en controle voor inhoud 0 voor de overstroomcontainer zou dan, gezien afronding van tijdstap, vanaf tijdstip 27 een kans van nul moeten geven voor de inhoud kleiner gelijk aan nul. Een twee dimensionale weergave van deze kansverdeling gemaakt door FST is hieronder weergegeven in figuur 16. Uit deze grafiek is duidelijk af te lezen dat de buffer begint te overstromen tussen tijdstip 26 en 27, waaruit blijkt dat modeluitbreiding in dit geval correct functioneert als equivalent van een continue test pijl.



Figuur 16: Doorsnee van figuur 15 met FST bij een volle bufferinhoud. Het sprongpunt geeft het tijdstip waarop er water binnenkomt en de overstroming begint.

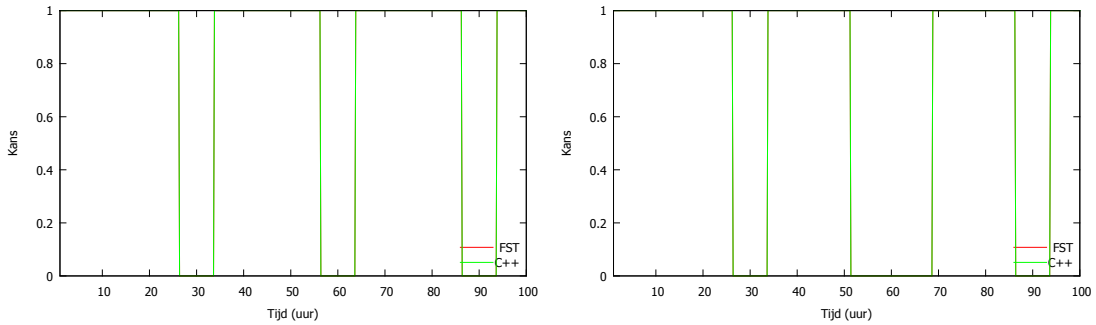
6.4 Validatie basismodel

Om te bepalen of de simulatie een goede afspiegeling is van het model dat is opgesteld is er validatie nodig. De validatie van de simulatie wordt gedaan door de resultaten hiervan te vergelijken met de resultaten van de Fluid Survival Tool. Hierbij gaan we ervan uit dat FST correct functioneert. Door met beide programma's hetzelfde Petrinet met gelijke begincondities en kansverdeling voor de stochastische transitie te analyseren of simuleren, kunnen de verkregen resultaten worden vergeleken en kunnen de resultaten van het simulatie programma gevalideerd worden. Hierbij moet gekeken worden of het BTI de waarden van de FST uitkomsten omvat en hoeveel runs nodig zijn om dit te bewerkstelligen.
Allereerst wordt hierbij gekeken naar een deterministisch scenario. Wanneer er geen sprake is van stochasticiteit zouden beide resultaten ook bij een enkele run overeen moeten komen. Hierna wordt een stochastisch scenario vergeleken. Het is hierbij de vraag welke kansverdelingen voor de reparatietijd, met welk aantal runs, een goed resultaat geven. Indien de verkregen resultaten uit het programma met een simulatie van een erg groot aantal runs niet teveel afwijken van de analyse van FST kan er geconcludeerd worden dat het programma naar behoren werkt.

6.4.1 Deterministische vergelijking

Allereerst is een deterministische situatie bekeken. Dit is gedaan voor een reparatietijd van 5 uur en van 10 uur. Wanneer er geen stochastische transities aanwezig zijn kunnen beide programma's namelijk op ieder tijdstip exact berekenen wat de inhoud van de buffer is. Deze uitkomsten geven de mogelijkheid de programma's goed te vergelijken. Deze vergelijking is gedaan door een willekeurige inhoud $c = 50.000m^3$ te kiezen en op ieder tijdstip de kans te berekenen dat de

inhoud van de buffer kleiner of gelijk aan c is. Met beide programma's kan vervolgens een grafiek worden geconstrueerd, waarin de kans tegen de tijd wordt uitgezet. In figuur 17 zijn voor twee verschillende reparatietijden de grafieken van beide programma's in dezelfde figuur geplot. In beide gevallen is te zien dat de grafieken geheel samenvallen. We kunnen dus concluderen dat in het deterministische geval de programma's met elkaar overeenkomen.



(a) Vergelijking met reparatietijd van 5 uur.

(b) Vergelijking met reparatietijd van 10 uur.

Figuur 17: Vergelijking tussen FST en het C++ programma met deterministische reparatietijden. Beide grafieken komen geheel overeen.

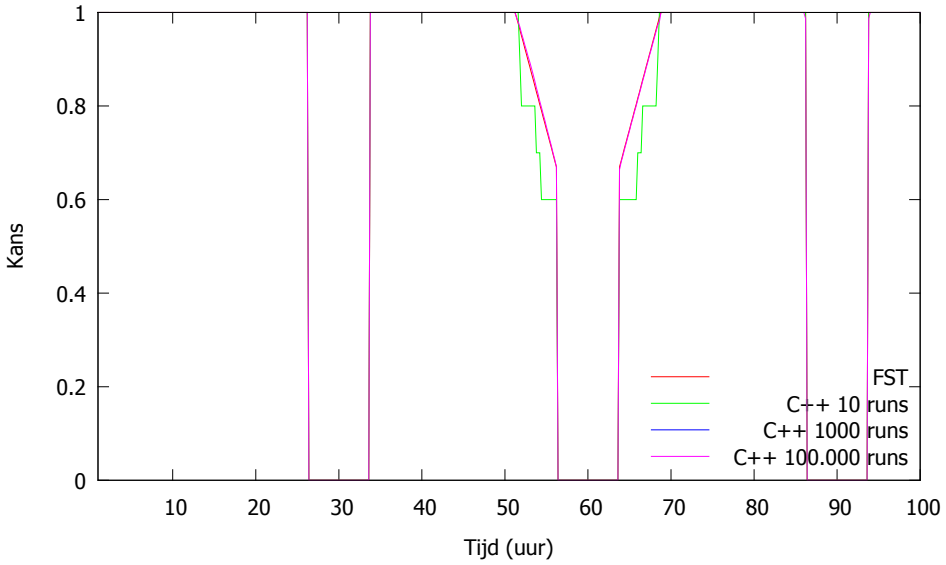
6.4.2 Stochastische vergelijking

Nu vastgesteld is dat het simulatieprogramma goed werkt bij deterministische scenario's, kan onderzocht worden wat er gebeurt wanneer stochasticiteit een rol speelt. Bij een kansverdeling met weinig spreiding zou de simulatie nog steeds een goede benadering moeten geven en zou door niet al te veel runs van het programma goed benaderd moeten worden. Hoe groter de spreiding en daarbij de onzekerheid wordt, hoe meer runs waarschijnlijk nodig zijn om een goede benadering te krijgen. Om deze reden is eerst een vergelijking uitgevoerd met een uniforme verdeling van de reparatietijd, waarbij de breedte is gevarieerd, hoe smaller de uniforme verdeling des te minder de spreiding. Vervolgens is gekeken hoe het gedrag is bij een exponentiële verdeling van de reparatietijd.

Uniforme verdeling Allereerst vergelijken we de programma's voor een uniform verdeelde reparatietijd. Hierbij bekijken we drie verschillende gevallen, waarbij we de uniforme verdeling telkens breder maken, zodat er sprake is van een grotere variantie. Net als in het deterministische geval bekijken we de kansdichtheid voor een bufferinhoud van $50.000m^3$.

In het eerste geval, dat te zien is in figuur 18 is gekeken naar een uniforme verdeling op het interval $(0, 10)$. In dit geval is de variantie van de reparatietijd gelijk aan 8,33. [4] Er is te zien dat in dit geval 1000 runs al voldoende is om FST met een betrouwbaarheid van 95% te benaderen.

[4] Hierbij is gebruik gemaakt van de standaardformule voor de uniforme verdeling: als $X \sim uniform(a, b)$ dan $var(X) = \frac{(b-a)^2}{12}$.



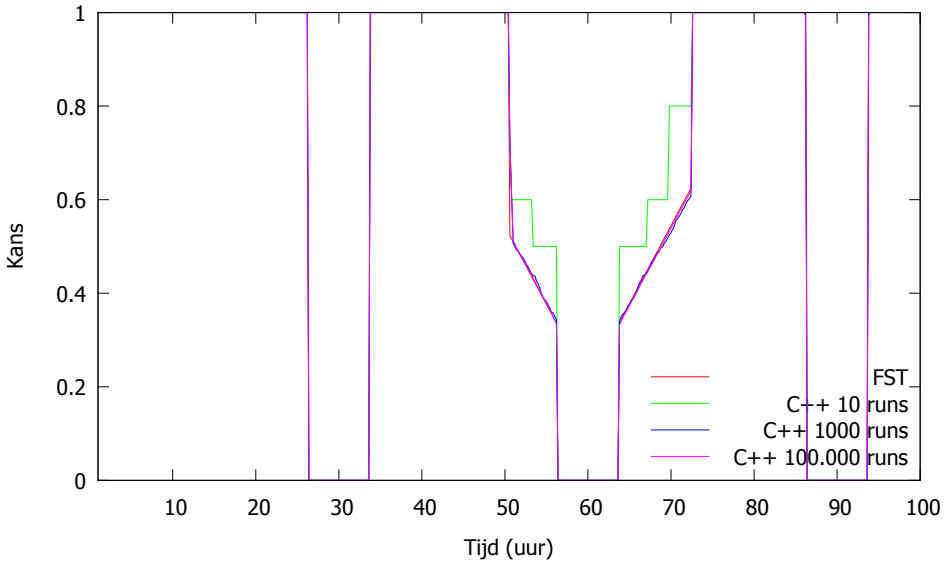
Figuur 18: Vergelijking tussen FST en simulatie met een reparatietijd $\sim Uniform(0, 10)$.

Het tweede geval dat is gesimuleerd, gebruikt voor de reparatietijd een uniforme verdeling op het interval $(0, 20)$, wat een variantie van 33,33 geeft. In figuur 19 is af te lezen dat in dit geval 1000 runs een minder goede benadering geeft dan bij een uniform(0,10) verdeling. Echter geeft 1000 runs nog steeds een betrouwbaarheid 95%. De enige afwijking hierbij is dat het sprongpunt op $t = 50$ bij FST een tijdstip eerder plaatsvindt dan bij C++. Dit komt omdat FST de tijd laat beginnen bij $t = 0.02$, terwijl C++ begint bij $t = 0$. Door afrondingsverschillen kan het daarom voorkomen dat een sprongpunt net een tijdstip eerder of later plaatsvindt.

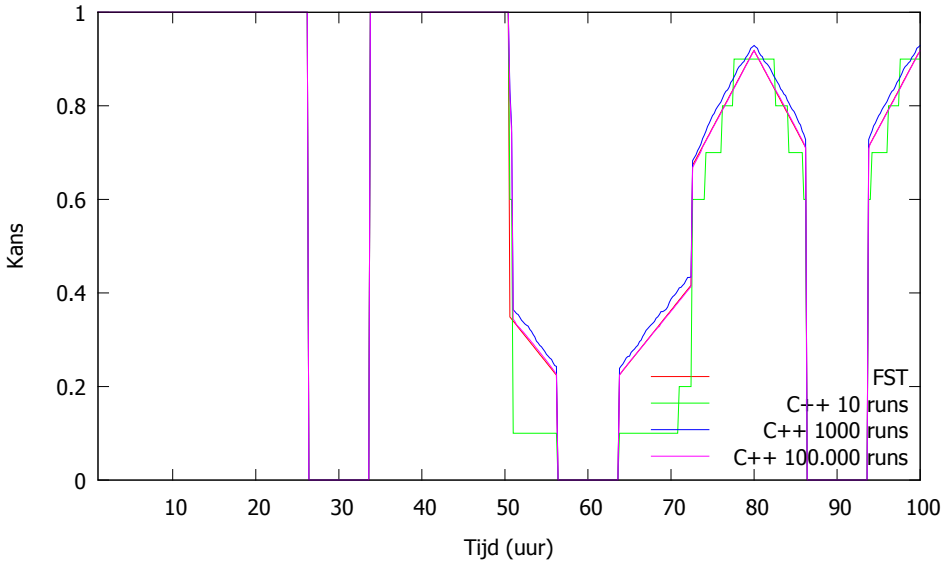
Als laatst is een uniforme verdeling op het interval $(0, 30)$ voor de reparatietijd bekeken, deze heeft dus een variantie van 75 uur. Bij hetzelfde aantal runs als bij voorgaande gevallen is in dit geval de benadering een stuk minder goed. Dit is te zien in figuur 20. In dit geval zijn er 100.000 runs nodig om een betrouwbaarheid van 95% te verkijgen. De bijbehorende betrouwbaarheidsintervallen zijn te zien in de bijlage in figuur 27 en 28. In dit geval is weer de enige afwijking het sprongpunt dat verschilt door een afwijkend gebruik van t.

Bovenstaande simulaties bevestigen dat een kansverdeling met een grotere variantie meer runs vereist voor een goede convergentie. Zelfs bij een uniforme kansverdeling met een variantie van 75 uur, is het voldoende om 100.000 runs te simuleren voor een betrouwbaarheid van 95% te bereiken wat betreft de voorspelling van de hoeveelheid water in de buffer. Deze simulatie kan bovendien in ongeveer 30 seconden uitgevoerd worden[5]. De analyse met FST heeft op eenzelfde machine slechts ongeveer 10% van de rekentijd nodig.

[5]Met een Intel core i5 processor.

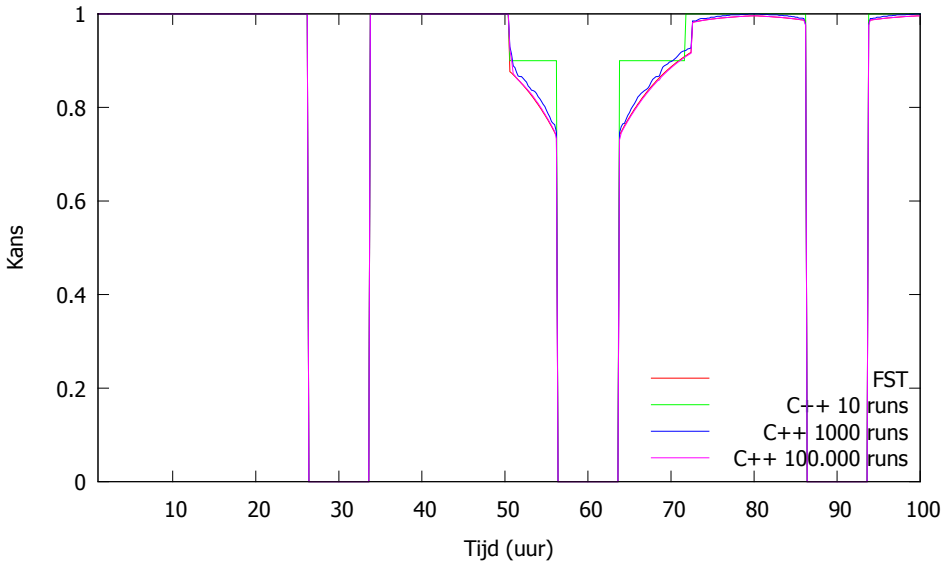


Figuur 19: Vergelijking tussen FST en simulatie met een reparatietijd $\sim Uniform(0, 20)$.



Figuur 20: Vergelijking tussen FST en simulatie met een reparatietijd $\sim Uniform(0, 30)$.

Exponentiële verdeling Naast de uniforme is ook de exponentiële verdeling bekeken. Wederom is de breedte van de kansverdeling en de hoeveelheid runs gevarieerd, om ook voor deze verdeling de werking van het programma te controleren. Er is gekozen om te simuleren met soortgelijke verwachtingswaardes als bij de uniforme verdeling. De exponentiële verdeling heeft voor dezelfde verwachtingswaarde een grotere variantie dan de uniforme verdeling en daarom zou het waarschijnlijk meer runs vereisen om de correcte kansverdeling te benaderen. In figuur 21 is het resultaat van de simulatie met de exponentiële verdeling met parameter 0.2, oftewel verwachtingswaarde 5. Dezelfde verwachtingswaarde als de uniforme verdeling op (0,10). In dit geval is de variantie 25 en is 1000 runs genoeg om te voldoen aan de gewenste betrouwbaarheid. [6].

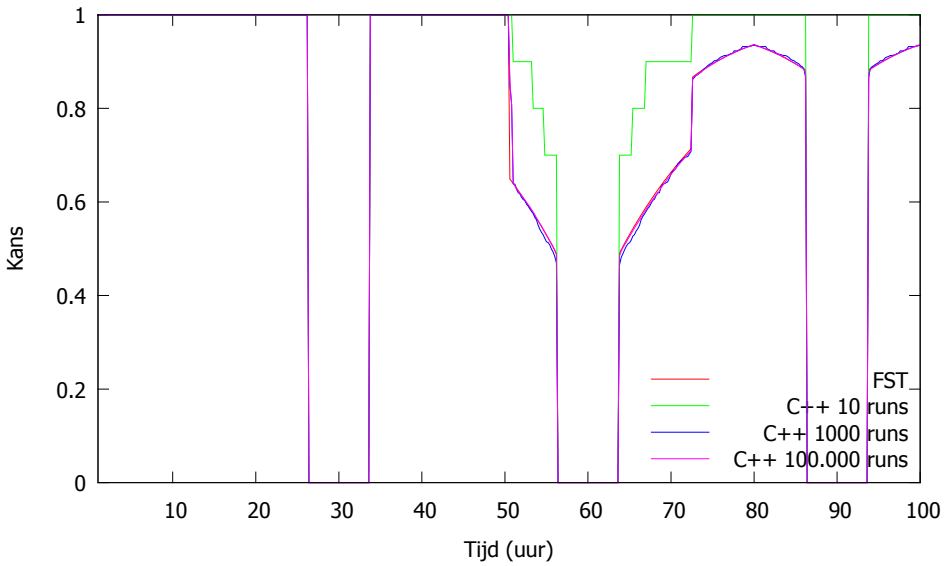


Figuur 21: Vergelijking met reparatietijd $\sim$ *Exponentieel*(0.2).

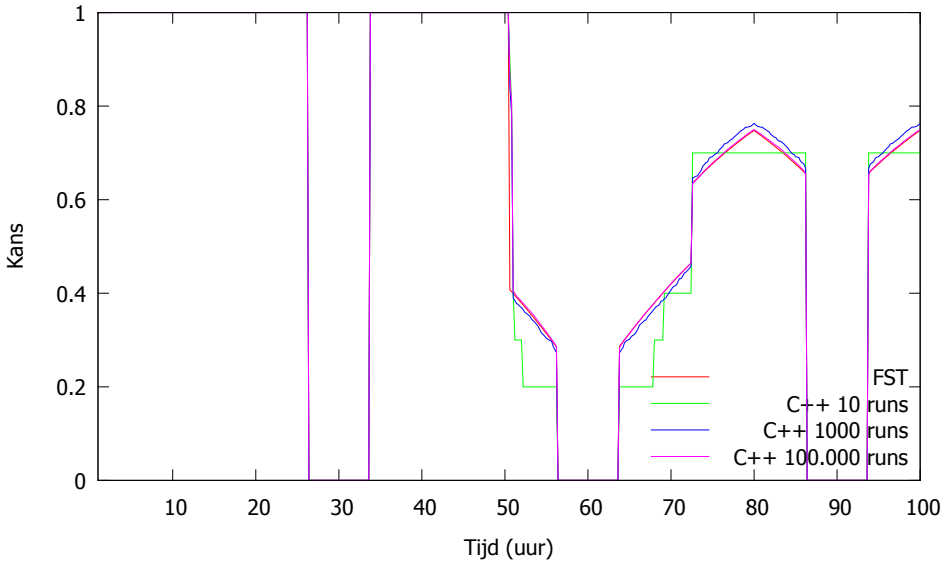
In figuur 22 is de benadering met parameter 0.1 weergegeven. Net na $t = 50$ is wederom een voorbeeld te zien van het verschil in de tijdstap. Verder geeft ook in dit geval 1000 runs al een goede benadering. Er is echter nog niet te zien dat deze benadering van 1000 runs slechter is dan de benadering met parameter 0.2. Dit is wel duidelijk te zien bij een nog grotere variantie met parameter 0.05 in figuur 23. Hier is te zien dat de blauwe lijn meer van verdeling van de FST afwijkt dan in bovenstaande gevallen. Een simulatie met 100.000 runs komt ook in dit geval erg goed in de buurt.

Vanwege de grotere variantie van de exponentiële verdeling ten opzichte van de uniforme verdeling is de verwachting dat er meer runs nodig zijn om een betrouwbaarheid van 95% te verkrijgen. Uit figuur 29 in de bijlage blijkt echter dat met parameter 0.05 en 1000 simulatieruns al 95% betrouwbaarheid bereikt wordt. Dit in tegenstelling tot de uniforme verdeling waar 100.000 runs nodig waren om deze betrouwbaarheid te bereiken. De benadering van de exponentiële verdeling in het C++ programma lijkt accurater dan de benadering van de uniforme verdeling.

[6] Exponentiële verdeling met parameter λ heeft een verwachtingswaarde van $\frac{1}{\lambda}$ en de variantie is $\frac{1}{\lambda^2}$



Figuur 22: Vergelijking met reparatietijd $\sim$ *Exponentieel*(0.1).

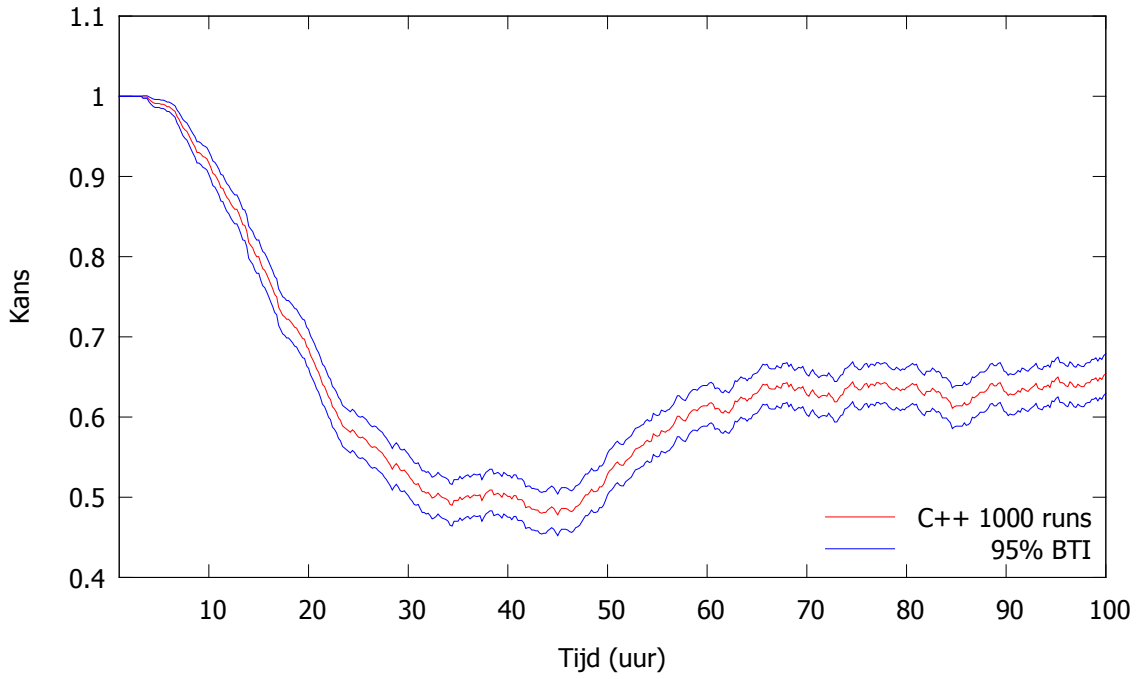


Figuur 23: Vergelijking met reparatietijd $\sim$ *Exponentieel*(0.05).

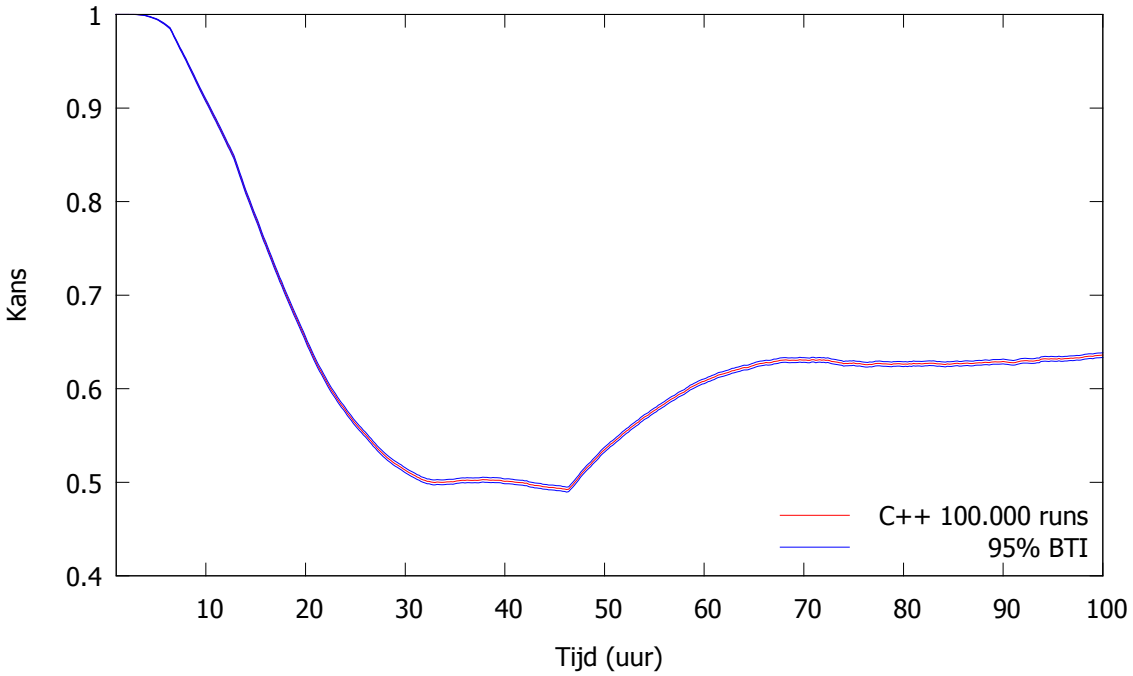
6.5 Validatie bij meerdere stochastische transities

De basis van de programmering is nu voor een enkele kansverdeling gevalideerd . Het doel van het programma is echter het geven van een benadering van een netwerk dat in de huidige versie van de FST niet geanalyseerd kan worden. Een netwerk waarin meerdere stochastische gebeurtenissen plaats kunnen vinden is hier een voorbeeld van.

Om ook te controleren of het model een goede benadering geeft voor dergelijke netwerken, is het basismodel uitgebreid. Hiervoor is gesimuleerd met de wisselingen van de aanvoer als extra stochastische events. Bovendien kunnen deze transities meerdere malen vuren. Voor de wisseling van droog weer naar regen is een uniforme verdeling op het interval $(0, 40)$ gebruikt en voor de wisseling van regen naar droog weer is de verdeling uniform $(0, 20)$. Verder is de reparatietijd in deze situatie exponentieel verdeeld met parameter 0.1. In figuur 24 is te zien dat met deze drie stochasten en slechts 1000 runs de verkregen kansen nog vrij onzeker zijn en er sprake is van een breed betrouwbaarheidsinterval. Wanneer meer runs worden gedaan heeft dit tot gevolg dat de kansverdeling gladder wordt en het betrouwbaarheidsinterval smaller. In figuur 25 is te zien dat ook bij drie stochasten die meerdere malen kunnen vuren 100.000 runs voldoende is om de kansverdeling te laten convergeren. De computatiesnelheid blijft in dit geval gelijk aan de computatietijd bij één stochast, ongeveer 30 seconden.



Figuur 24: Simulatie met meerdere stochasten, 1000 runs



Figuur 25: Simulatie met meerdere stochasten, 100.000 runs

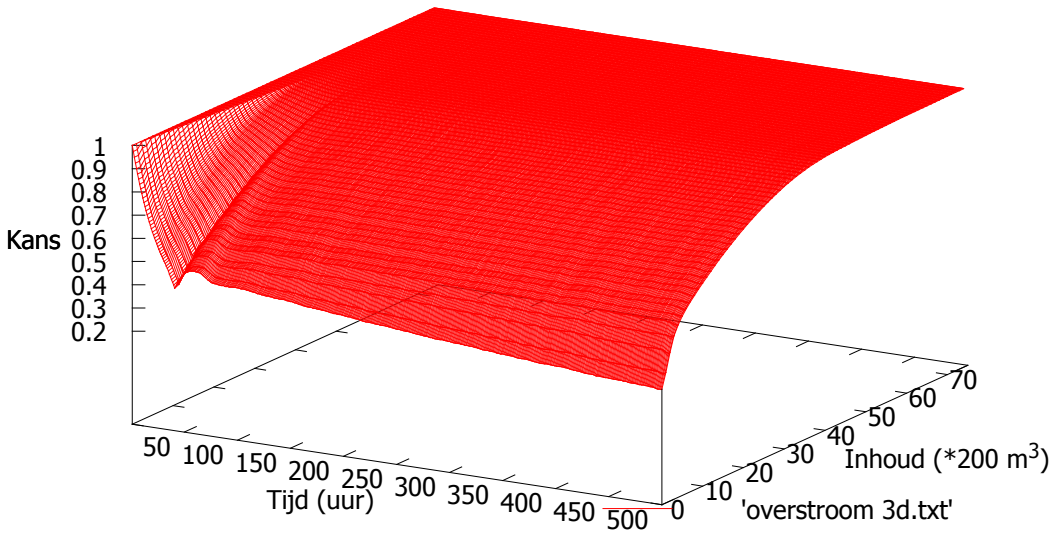
6.6 Voorbeeldsituatie

Uiteindelijk is het van belang om te weten hoe groot de kans is dat een deel van Enschede onder water komt te staan. Hiervoor moet de kans op overstroming van de buffer van de waterzuiveringsinstallatie bepaald worden. Om dit te onderzoeken is een simulatie gedaan over een tijdsduur van 500 uur. Hierbij is de exponentiële verwachting dat gemiddeld na 1000 uur de stroom uitvalt of het gemaal op andere wijze stuk gaat. Verder zijn de stochastische perioden van regen en droog weer uniform verdeeld met verwachting van respectievelijk 10 en 20. Figuur 26 is een weergave van de kansen voor de inhoud van de buffer uitgezet tegen de tijd.

Om 500 uur te simuleren zijn tijdstappen van een uur gebruikt. Kleinere tijdstappen zou meer rekentijd betekenen en het programma heeft nog moeite met een grote dataoutput. Wel betekent een grotere tijdstap een grotere afrondingsfout voor de stochastische events. Door deze aanpassing blijft de simulatietijd voor 100.000 runs rond de 30 seconden, net als bij de simulatie van 100.000 runs voor een simulatieduur van 100 uur met tijdstap van 0.2, want het aantal tijdstappen blijft 500. In grafiek 26 is te zien dat in de eerste tijdstappen de kans over de tijd nog erg schommelt maar deze na meerdere uren convergeert naar een vaste kans bij iedere inhoud. Dit komt omdat de onzekerheid van de stochastische gebeurtenissen zich opstapelt wanneer er meer plaatsvinden en er steeds minder duidelijkheid is op elk tijdstip. In het begin is er bij een bepaalde tijdstap nog meer zekerheid of een event wel of niet heeft plaatsgevonden, terwijl deze kans later over de tijd convergeert naar de limietverdeling van deze gebeurtenissen.

De kans dat de bufferinhoud kleiner of gelijk is aan de maximale inhoud 15.000 m^3 komt erg dicht in de buurt van 1, maar wijkt nog enigszins af. Voor elke tijdstap bestaat een kleine kans

dat de buffer overstroomt, de kans dat minstens één moment in 500 uur de buffer overstroomt is afhankelijk van al die mogelijke momenten van overstromen. Uit de simulatie blijkt dat de kans op minstens één overstroming gelijk is aan $10,7\%$. Echter de hoeveelheid water die in dit geval totaal overstroomt is zodanig klein dat deze geen problemen zou opleveren.



Figuur 26: Simulatie van 100.000 runs voor een langere tijdsperiode van 5000 uur. De y-as geeft het bufferniveau maal 5, de z-as geeft vervolgens de kans dat de inhoud kleiner of gelijk aan deze waardes is.

7 Discussie

Gedurende het modelleerproces zijn aannames en simplificaties gedaan die mogelijk tot gevolg hebben dat de resultaten niet geheel accuraat zijn. In dit hoofdstuk worden deze daarom kritisch besproken en wordt het effect hiervan bekeken.

Model In het uiteindelijke model zijn aannames gemaakt om het simuleren eenvoudiger te maken, zodat in eerste instantie de werking van het programma gecontrolleerd kan worden.

Een van de gebruikte aannames is dat regen discreet is, er is telkens een deterministisch interval wel of geen sprake van regen. Op deze manier kan namelijk makkelijk onderzocht worden wat het effect is van vrij heftige regen in combinatie met een kapotte pomp, aangezien er dan kans is op overstroming van de buffer. Wanneer echter een realistische analyse van de waterzuivering wordt gedaan is het beter het regenproces continu te modelleren, aangezien dit beter overeen komt met de werkelijkheid.

Verder is de tijd totdat de pomp kampot gaat deterministisch gekozen om het effect op het model te kunnen onderzoeken. De reparatietijd van de pomp is vervolgens als onzeker beschouwd, omdat de tijd dat de pomp kapot is een situatie is waar de kans op overstroming aanwezig is. Bij simulatie zijn voor de reparatietijd verschillende kansverdelingen gekozen om de werking van het programma te controleren. De keuze van deze verdelingen is echter niet gebaseerd op een realistische reparatieperiode. Voor een realistische situatieschets van de waterzuivering moet deze kansverdeling zorgvuldig worden gekozen en zou gekeken kunnen worden naar rare event simulation.

Simulatie Na validatie en verificatie blijkt dat de analyse van de Fluid Survival Tool goed te benaderen is door middel van Discrete Event Simulation. Echter zijn er wel nog enkele punten die enigszins afwijken.

Allereerst is het programma niet voor precies dezelfde tijdstippen vergeleken, maar was een afwijking van 0.02 seconden aanwezig. FST begint de simulatie namelijk op tijdstip 0.02, terwijl C++ gewoon op tijdstip 0 begint. Hierdoor worden twee net iets verschillende tijdstippen vergeleken en kunnen de sprongpunten een tijdstap afwijken. De vergelijking is hierdoor niet helemaal kloppend, ook al zijn de versprongen tijdstappen te herleiden uit de grafieken. Een precieze vergelijking heeft natuuurlijk de voorkeur. Het C++ programma is hier niet op aangepast aangezien dit een gecompliceerde verandering van het programma zou betekenen en het beginnen van de simulatie op tijdstip nul meer voor de hand ligt. Het zou eventueel opgelost kunnen worden door tijdstappen van 0.02 te gebruiken, maar dit kost veel extra rekentijd.

In de keuze van tijdstappen is een afweging gemaakt tussen snelheid en precisie. Het gebruik van discrete tijdstappen in de programmering verplichten het programma namelijk tot afronden. Met de gekozen implementatie worden de stochastische eventtijden naar boven, naar de volgende tijdstap, afgerond. Als in een simulatie meerdere stochastische eventtijden zouden plaatsvinden wordt er meerdere keren afgerond en kan een grotere fout ontstaan. Het gebruik van kleinere tijdstappen zou een kleinere afrondingsfout opleveren, maar is duurder qua rekentijd. Ook moet hiervoor eerst de manier van output genereren aangepast moeten worden, aangezien de output van het simulatieprogramma momenteel nog geen al te groot formaat aan kan nemen.

Bij het simuleren van een kansverdeling is er een onzekerheid in de benadering van de kansverdeling, door het aantal runs te vergroten kan de onzekerheid flink worden verminderd. Hoeveel runs er nodig zijn om een gewenste benadering te doen hangt af van de spreiding van de kansverdeling. In het geval van een groot netwerk, meer meerdere stochasten met een vrij grote spreiding kan het aantal benodigde runs flink oplopen.

8 Conclusie

De rioolwaterzuiveringsinstallatie is gemodelleerd in de vorm van een hybride Petrinet. Bij deze modellering is het mogelijk zowel discrete als continue factoren van de installatie verwerkt. Hierdoor kan de hoeveelheid water die aangevoerd wordt fluctueren, kunnen verschillende onderdelen van de installatie kapot gaan en is een overstroming mogelijk.

Het gedrag van dit model kan bekeken worden met behulp van een C++ simulatieprogramma voor Petrinetten, dat een kansverdeling geeft voor de inhoud van een gewenste plaats. Met behulp van de al bestaande Fluid Survival Tool is verificatie uitgevoerd, waaruit blijkt dat de simulatie een goede benadering geeft van het gedrag van het Petrinet. In deterministische gevallen komen de uitkomsten van de simulatie geheel overeen met FST. Door testen met een uniforme en een exponentiële verdeling is vervolgens de invloed van stochasticiteit onderzocht. Hieruit bleek dat de resulaten van de simulatie binnen het gewenste 95% betrouwbaarheidsinterval vallen, wanneer voldoende runs gedaan worden.

In eenvoudige gevallen zijn er slechts 1000 runs nodig om een goed resultaat te bereiken, welke binnen enkele seconden uitgevoerd kunnen worden. Echter wanneer de kansverdeling breder wordt zijn meer runs nodig om het simulatieprogramma te laten convergeren. Wanneer er gebruik wordt gemaakt van één stochast die eenmaal kan vuren is het bij de onderzochte gevallen voldoende om 100.000 runs te doen. Deze simulatie kan nog steeds binnen een minuut plaatsvinden.

Situaties met meerdere stochastische transities die ook meerdere malen kunnen vuren zijn gecontroleerd aan de hand van convergentie naar een gladde kansverdeling en de breedte van het betrouwbaarheidsinterval. Bij een simpel model met meerdere kansverdeling is te zien dat minder snel convergentie wordt bereikt. Echter zijn 100.000 runs nog steeds voldoende voor een goede betrouwbaarheid en is ook hier de simulatie binnen een minuut uit te voeren.

Het simulatieprogramma lijkt geschikt om uitspraken te doen over de risicofactoren van de waterzuivering. Voor een realistisch beeld van deze risico's kan een uitgebreid Petrinetten met meerdere stochastische factoren gesimuleerd worden. We zijn er echter niet aan toegekomen om deze risico's te onderzoeken.

9 Aanbevelingen

Aangezien is gebleken dat het programma correct functioneert, kan verder in gebruik worden genomen om Petrinetten te simuleren en analyseren. Voor het gebruik hiervan hebben we nog enkele aanbevelingen die in verder onderzoek uitgevoerd zouden kunnen worden.

Het programma in zijn huidige vorm kan gebruikt worden om de daadwerkelijke analyse aan de waterzuivering te doen. De aanbeveling hierbij is om ook gebruik te maken van de voorgestelde uitbreidingen. Het is dan interessant om te kijken welke gebeurtenissen een overstroming tot gevolg hebben en of de kans hierop significant is. Door op verschillende tijdstippen verschillende delen van de installatie kapot te laten gaan, kan onderzocht worden welk onderdeel het gevoeligst is voor sabotage of incorrect functioneren en op welk tijdstip dit het meeste effect zou hebben. Verder kan met behulp van de genoemde risicofactoren onderzocht worden wanneer er een grote kans is op een overstroming en welke maatregelen hiertegen effect zouden hebben. Het is hierbij interessant om te bekijken of deze maatregelen ook een significante verbetering opleveren.

Op het moment dat het programma wordt gebruikt voor analyse van risicofactoren is het belangrijk om de gebruikte parameters van het model goed te onderzoeken, zodat deze realistisch gekozen kunnen worden. Hieronder vallen bijvoorbeeld de verdeling van de reparatietijd, de hoeveelheden aangevoerd water en de grootte van de buffer. Bij de verificatie is namelijk gebruik gemaakt van tamelijk willekeurige parameters. Het is vooral belangrijk de verdeling van de reparatietijd goed te onderzoeken en te kiezen.

Wanneer het lang droog weer is geweest, hoopt er veel slib op in het riool. Hierdoor krijgt de waterzuiveringsinstallatie bij de eerstvolgende regenbui extra veel slib te verwerken. Het is daarom interessant om te kijken hoe de installatie het best met dit scenario om kan gaan. Een mogelijke oplossing is dat het riool in droge tijden regelmatig doorgespoeld wordt om zo het ophopen van slib tegen te gaan. Om ook dit probleem met het model te onderzoeken zullen nog enkele uitbreidingen toegevoegd moeten worden.

Daarnaast kan het programma gebruiksvriendelijker gemaakt worden. Op dit moment is het slechts inzetbaar voor het hier ontwikkelde model. Door de implementatie algemener te maken, zou het makkelijker moeten zijn om het model op een eenvoudige manier aan te passen en zelfs ieder willekeurig Petrinet ermee te simuleren. Nu is hiervoor nog programmeerkennis nodig.

Ook kan het interessant zijn om een groot model met meerdere kansverdelingen op te delen in kleinere delen met allen slechts één stochastische transitie. Vervolgens zou de analyse van een klein stuk van het model met FST kunnen worden gedaan en kunnen deze exacte resultaten met behulp van het simulatie programma aan elkaar gekoppeld worden. Op deze manier kunnen grote modellen met veel onzekerheid op een meer exacte manier benaderd worden. Om dit toe te kunnen passen zijn wel nog enkele toevoegingen aan het programma nodig.

10 Referenties

[1] URL `https://code.google.com/p/fluid-survival-tool/`. [3, 19]

[2] W. Albers. Wiskundige statistiek, September 2010. [24]

[3] G. Ciardo, D. Nicol, and K.S. Trivedi. Discrete-event simulation of fluid stochastic petri nets. *IEEE Transactions on Software Engineering*, 2(25):207–217, 1999. doi:10.1109/32.761446. [11]

[4] R. David and H. Alla. On hybrid petri nets. *Discrete Event Dynamic Systems*, 11(1):9–40, 2001. doi:10.1023/A:1008330914786. [9]

[5] H. Ghasemieh, A. Remke, B. Haverkort, and M. Gribaudo. Region-based analysis of hybrid petri nets with a single general one-shot transition. page 139, 2012. [9]

[6] C. Girault and R. Valk. *Petri Nets for Systems Engineering: A Guide to Modelling, Verfication and Applications*. Springer-Verlag, 2003. [8]

[7] M. Gribaudo and A. Remke. Hybrid petri nets with general one-shot transitions. [10]

[8] M. Gribaudo and A. Remke. Hybrid petri nets with general one-shot transitions for dependability evaluation of fluid critical infrastructures. *High-Assurance Systems Engineering (HASE), 2010 IEEE 12th International Symposium on*, 2010. doi:10.1109/HASE.2010.27. [3, 9, 19]

[9] M. Gribaudo, M. Sereno, A. Horvath, and A. Bobbio. Fluid stochastic petri nets augmented with flush-out arcs: Modelling and analysis. *Discrete Event Dynamic Systems*, 11(1):97–117, 2001. doi:10.1023/A:1008339216603. [14]

[10] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77 (4):541–580, 1989. doi:10.1109/5.24143. [3, 7]

[11] C.A. Petri. *Kommunikation mit Automaten*. PhD thesis, Technische Univarsitat Darmstadt, 1962. [7]

[12] W. Reisig. *Petrinetze, Eine Einfuhrung*. Springer, 1982. ISBN 354011478. [7]

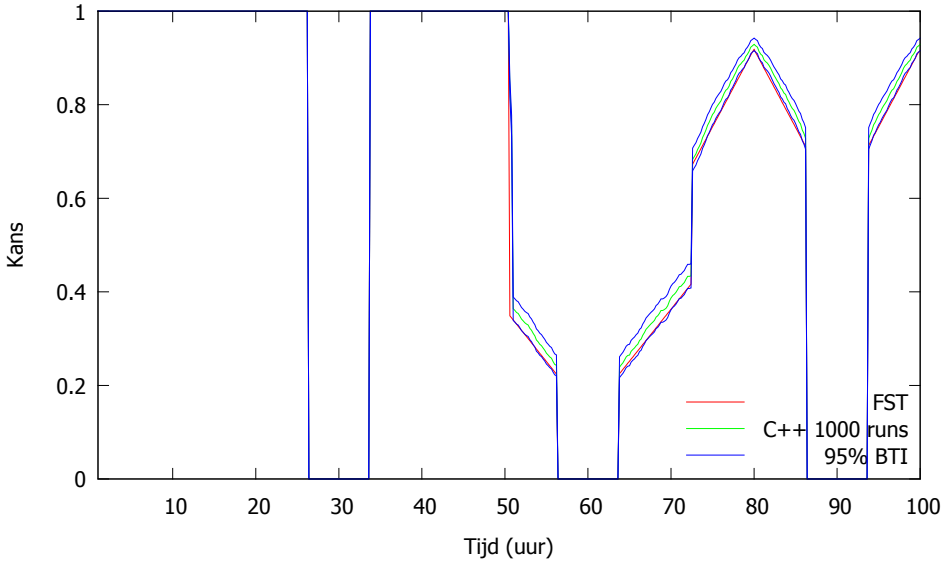
11 Symbolenlijst

Symbool	Beschrijving	Verzameling van
$\mathcal{P}_D$	Discrete plaatsen	$\mathcal{P}$
$\mathcal{P}_C$	Continue plaatsen	$\mathcal{P}$
$\mathcal{T}_C$	Continue transities	$\mathcal{T}$
$\mathcal{T}_I$	Directe transities	$\mathcal{T}$
$\mathcal{T}_D$	Deterministische transities	$\mathcal{T}$
$\mathcal{T}_S$	Stochastische transities	$\mathcal{T}$
$\mathcal{T}_F$	Discrete transities $\mathcal{T}_I \cup \mathcal{T}_D \cup \mathcal{T}_S$	$\mathcal{T}$
$\mathcal{A}^D$	Discrete input of output pijlen	$((\mathcal{P}_D \times \mathcal{T} \backslash \mathcal{T}_F) \cup (\mathcal{T} \backslash \mathcal{T}_F \times \mathcal{P}_D))$
$\mathcal{A}_C$	Continue input of output pijl	$((\mathcal{P}_C \times \mathcal{T}_C) \cup (\mathcal{T}_C \times \mathcal{P}_C))$
$\mathcal{A}_T$	Test pijl	$(\mathcal{P}_C \times \mathcal{T})$

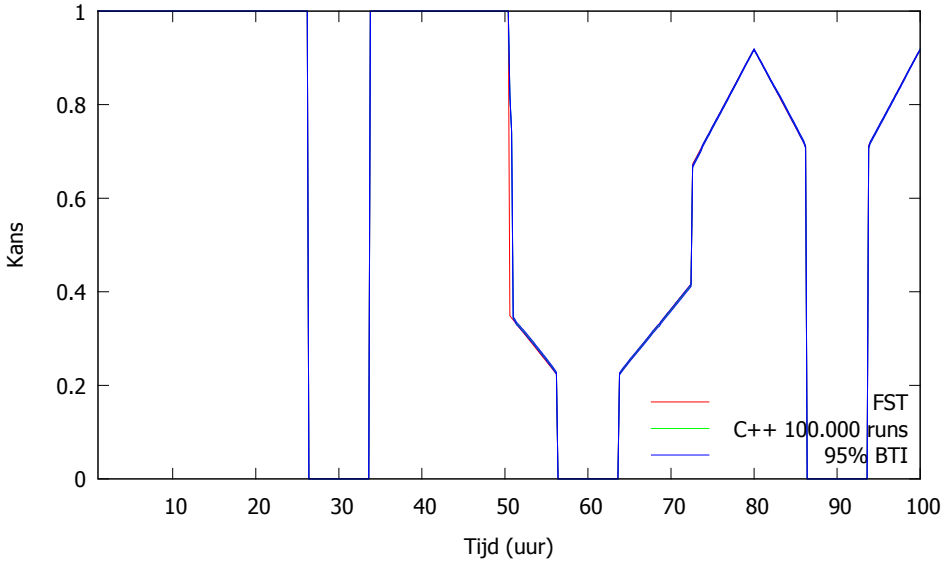
Symbool	Beschrijving	Eenheid
$\phi_b^{\mathcal{P}}$	bovengrens continue plaats $\mathcal{P}^c$	$\mathcal{P}^c \to \mathbb{R}^+ \cup \infty$
ϕ_w^T	gewicht van een transitie $\mathcal{T}$	$\mathcal{T} \setminus \mathcal{T}_C \to \mathbb{R}^+$
ϕ_p^T	prioriteit van een transitie $\mathcal{T}$	$\mathcal{T} \setminus \mathcal{T}_C \to \mathbb{N}$
ϕ_d^T	tijd voordat de transitie $\mathcal{T}_D$ plaatsvindt	$\mathcal{T}_D \to \mathbb{R}$
ϕ_f^T	stroomsnelheid van continue transitie $\mathcal{T}_C$	$\mathcal{T}_C \to \mathbb{R}^+$
ϕ_g^T	continue kansverdeling voor stochastische transitie $\mathcal{T}_S$	
ϕ_w^T	gewicht van een pijl $\mathcal{A}$	$\mathcal{T} \setminus \mathcal{T}_C \to \mathbb{R}^+$
ϕ_p^T	prioriteit van een pijl $\mathcal{A}$	$\mathcal{T} \setminus \mathcal{T}_C \to \mathbb{N}$

Symbool	Beschrijving
$\hat{p}$	Schatter van de kansverdeling voor bufferinhoud
X	Totaal aantal succesen voor bepaalde bufferinhoud
n	Aantal runs van een simulatie
γ	Gewenste betrouwbaarheid

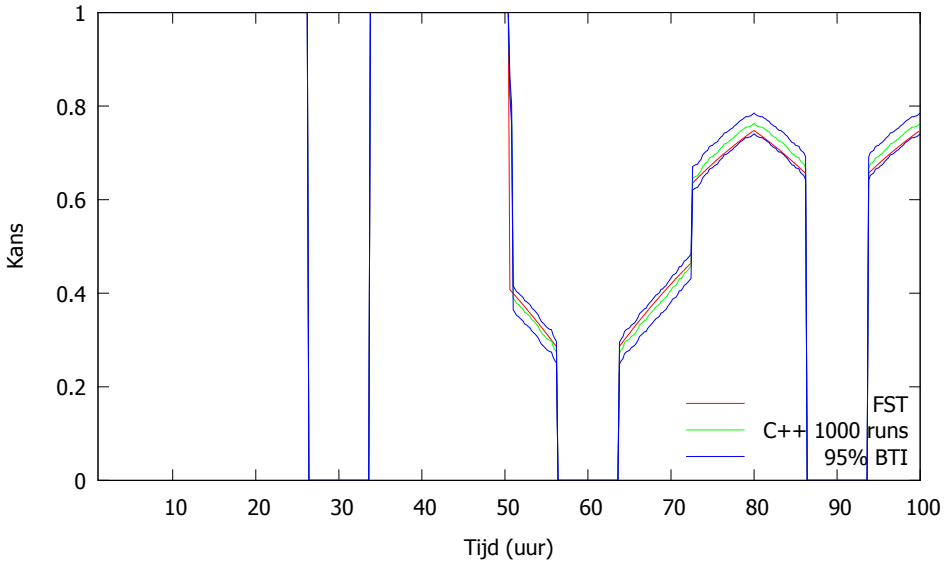
12 Bijlage



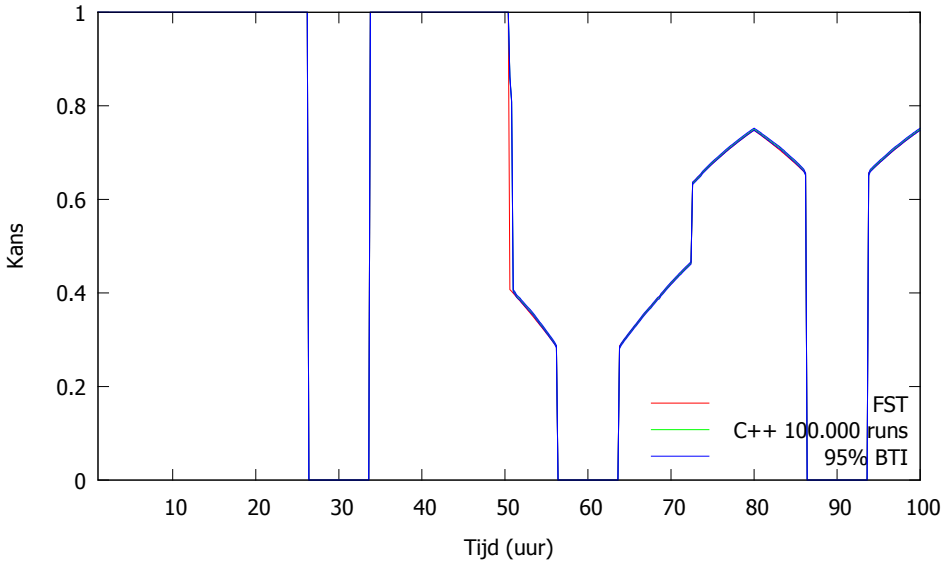
Figuur 27: Betrouwbaarheidsinterval van 95% bij een verdeling $\sim Uniform(0, 30)$ met 1000 runs.



Figuur 28: Betrouwbaarheidsinterval van 95% bij een verdeling $\sim Uniform(0, 30)$ met 100.000 runs.



Figuur 29: Betrouwbaarheidsinterval van 95% bij een verdeling $\sim Exp(0.05)$ met 1000 runs.



Figuur 30: Betrouwbaarheidsinterval van 95% bij een verdeling $\sim Exp(0.05)$ met 100.000 runs.