Inhoudsopgave

1. Inleiding

Het softwarebedrijf MathWorks is bezig het statistische computerpakket StatWorks te ontwikkelen. Op dit moment is de software voor dit pakket in de testfase, dat wil zeggen: de code is af, maar de software wordt nu getest op fouten. De klanten van MathWorks willen weten wanneer het pakket wordt vrijgegeven. Daarom wil de manager een schatting maken van de releasedatum die aan de klanten kan worden doorgegeven. Dit moet een tijdstip zijn waarop de software voldoende betrouwbaar is. Er moeten dan voldoende fouten uit de software zijn verwijderd, zodat de kans op het langdurig foutloos draaien van de software groot is. Ons doel van dit verslag is met behulp van kansrekening en statistiek een schatting te maken voor de releasedatum.

2. Een eenvoudig model voor de betrouwbaarheid van software

Op het moment dat er nog fouten in de software zitten, zullen deze bij het testen van de software tot *storingen* leiden. We zullen spreken van een storing als de software bij bepaalde input niet de gewenste output levert, of eventueel ook: als de uitvoering van een bepaalde opdracht te veel tijd in beslag neemt. Aldus gaat een testteam de software testen en worden tijdstippen geregistreerd waarop de storingen optreden.

We zetten als volgt een (nu nog eenvoudig) model op. Neem aan:

- Er zitten twee fouten in de software. Ieder van deze fouten leidt op den duur tot een storing.
- We beginnen de test op een tijdstip $t = 0$.
- De tijd wordt gemeten in CPU-seconden. Dat is een maat voor de tijd dat de computer bezig is met het uitvoeren van opdrachten.
- Na iedere storing wordt de CPU-tijd stilgezet en wordt eerst de betreffende fout verbeterd door een reparatieteam. Indien dan nog niet beide fouten zijn verbeterd, vervolgt het testteam het testen van de software. Daarbij loopt de CPU-klok weer.
- De eerste storing vindt plaats op tijdstip t_1 en de tweede op t_2 $(t_1 < t_2)$.
- Ieder van de twee storingen wordt veroorzaakt door precies één van beide fouten (niet door beide fouten).
- Tijdens de reparatie van een fout treden geen nieuwe fouten op.
- t_1 en t_2 zijn realisaties van de stochastische grootheden T_1 en T_2.
- Fout nr. 1 veroorzaakt op tijdstip T_1' en fout nr. 2 doet dat op tijdstip T_2'. Er geldt dan: $T_1 = \min\{T_1', T_2'\}$ en $T_2 = \max\{T_1', T_2'\}$.
- T_1' en T_2' zijn onafhankelijke en identiek verdeelde stochasten met een continue verdeling gegeven door dichtheid f en verdelingsfunctie F.

We kunnen nu de verdeling van T_1 en T_2 bepalen. We gebruiken de notaties F_{T_1}, F_{T_2}, f_{T_1} en f_{T_2} voor de marginale verdelingsfuncties en dichtheden van T_1 en T_2. We vinden:

$$F_{T_1}(t) = P(T_1 \le t) = P(\min\{T_1', T_2'\} \le t) = P\big(\{T_1' \le t\} \cup \{T_2' \le t\}\big) =$$
$$P(T_1' \le t) + P(T_2' \le t) - P\big(\{T_1' \le t\} \cap \{T_1' \le t\}\big) = 2P(T_1' \le t) - \big(P(T_1' \le t)\big)^2 =$$
$$2F(t) - F^2(t) = F(t)\,(2 - F(t)).$$

Dus:

$$f_{T_1}(t) = \frac{d}{dt} F_{T_1}(t) = 2F'(t)\,(1 - F(t)) = 2f(t)\,(1 - F(t)).$$

Op dezelfde manier volgt:

$$F_{T_2}(t) = P(T_2 \le t) = P(\max\{T_1', T_2'\} \le t) = P\big(\{T_1' \le t\} \cap \{T_2' \le t\}\big) = \big(P(T_1' \le t)\big)^2 = F^2(t).$$

Dus:

$$f_{T_2}(t) = \frac{d}{dt} F_{T_2}(t) = 2F(t)F'(t) = 2f(t)F(t).$$

Nu we de verdelingen van T_1 en T_2 weten, kunnen we de volgende drie interessante kansen uitrekenen, die iets zeggen over het aantal storingen dat heeft plaatsgevonden voor of op een bepaald tijdstip.

$$\begin{aligned}
P(\text{geen storingen voor } t) &= P(T_1 > t) = 1 - P(T_1 \le t) = F^2(t) - 2F(t) + 1 = (1 - F(t))^2. \\
P(\text{twee storingen voor } t) &= P(T_2 \le t) = F^2(t). \\
P(\text{één storing voor } t) &= 1 - P(\text{geen storingen voor } t) - P(\text{twee storingen voor } t) = \\
&\quad -2F^2(t) + 2F(t) = 2F(t)\,(1 - F(t)).
\end{aligned}$$

We kunnen nu een heel belangrijke stochast definiëren: $M(t)$ is het aantal storingen dat heeft plaatsgevonden voor of op tijdstip t. $M(t)$ is een discrete stochastische grootheid met waardenverzameling $W_{M(t)} = \{0, 1, 2\}$. Altijd geldt: $M(0) = 0$ en $M(\infty) = 2$, hetgeen direct volgt uit onze aannames. Uit de bovenstaande berekeningen voor het aantal storingen voor of op tijdstip t kunnen we nu direct voor willekeurige $t > 0$ de kansverdeling van $M(t)$ geven:

$$P(M(t) = m) = \left\{ \begin{array}{lll} (1 - F(t))^2 & \text{als} & m = 0 \\ 2F(t)\,(1 - F(t)) & \text{als} & m = 1 \\ F^2(t) & \text{als} & m = 2 \end{array} \right\} = \binom{2}{m} F^m(t)\,(1 - F(t))^{2-m}.$$

Conclusie: $M(t)$ heeft een binomiale verdeling met parameters 2 en $F(t)$. Er geldt dus onder meer: $E(M(t)) = 2F(t)$ en $Var(M(t)) = 2F(t)\,(1 - F(t))$.

We kijken nog eens naar de stochast $M(t)$ en merken de volgende relatie op tussen $M(t)$ en het koppel T_1 en T_2:

$$M(t) \geq i \quad \Leftrightarrow \quad T_i \leq t \qquad \text{voor } i \in \{1, 2\}.$$

We hebben deze relatie stilzwijgend al gebruikt bij de bepaling van de verdeling van $M(t)$.

Laat t_1 en t_2 nu twee willekeurige tijdstippen zijn, met $0 < t_1 < t_2$. De positieve reële rechte wordt hiermee verdeeld in drie disjuncte intervallen, namelijk $(0, t_1]$, $(t_1, t_2]$ en (t_2, ∞). Het aantal storingen dat plaatsvindt in deze intervallen zijn resp. $M(t_1)$, $M(t_2) - M(t_1)$ en $M(\infty) - M(t_2) = 2 - M(t_2)$. In het vervolg zullen we deze drie stochasten ook wel noteren met resp. X, Y en Z. We zijn nu geïnteresseerd in de kansverdeling van de vector $V := (X, Y, Z)$. Om die te bepalen hebben we een klein stukje theorie nodig.

Stel: we doen n onafhankelijke en gelijke experimenten. Elk experiment heeft k mogelijke uitkomsten en de i-de uitkomst treedt op met kans p_i. Laat N_i het aantal keer voorstellen dat uitkomst i optreedt tijdens de n experimenten. Dan heeft de vector $(N_1, \ldots, N_k)$ een *multinomiale verdeling* met parameters n en kansvector $(p_1, \ldots, p_k)$. Dat wil zeggen:

$$P(N_1 = n_1, \ldots, N_k = n_k) = \binom{n}{n_1, \ldots, n_k} p_1^{n_1} \cdots p_k^{n_k} = \frac{n!}{n_1! \cdots n_k!} p_1^{n_1} \cdots p_k^{n_k},$$

voor $(n_1, \ldots, n_k) \in \{0, 1, \ldots, n\}^k$ zodat $n_1 + \ldots + n_k = n$, en nul anders.

In ons geval zijn er twee fouten in de software aanwezig $(n = 2)$ en voor elke fout kunnen we het optreden van de bijbehorende storing zien als een experiment waarbij onafhankelijk één van de drie genoemde intervallen $(k = 3)$ wordt gekozen. Dus het aantal storingen dat in de drie intervallen optreedt, is multinomiaal verdeeld. De kansvector bepalen we door voor een storing na te gaan hoe groot de kans is dat deze in een bepaald interval optreedt. Voor $i \in \{1, 2\}$ hebben we:

$$\begin{aligned} P(T_i' \in (0, t_1]) &= P(T_1' \leq t_1) = F(t_1). \\ P(T_i' \in (t_1, t_2]) &= P\big(\{T_1' \leq t_2\} \setminus \{T_1' \leq t_1\}\big) = F(t_2) - F(t_1). \\ P(T_i' \in (t_2, \infty)) &= P(T_1' > t_2) = 1 - P(T_1' \leq t_2) = 1 - F(t_2). \end{aligned}$$

Dus de kansvector is $(F(t_1), F(t_2) - F(t_1), 1 - F(t_2)) =: (p_1, p_2, p_3)$. Merk op: $p_1 + p_2 + p_3 = 1$.

Conclusie: de stochastische vector $V = (X, Y, Z)$ heeft een multinomiale verdeling met parameters 2 en kansvector (p_1, p_2, p_3), dus:

$$P(V = (x, y, z)) = \frac{2}{x!y!z!} p_1^x p_2^y p_3^z,$$

voor $(x, y, z) \in \{0, 1, 2\}^3$ zodat $x + y + z = 3$, en nul anders.

We zullen nu nog enkele verdelingen en kansen bepalen die met de vector V te maken hebben. Allereerst bekijken we de drie marginalen behorende bij deze multinomiale verdeling. Omdat $X = M(t_1)$, en dus een bijzonder geval van de eerder gedefinieerde $M(t)$ is, zien we direct dat X een binomiale verdeling heeft met parameters 2 en $F(t_1) = p_1$. Omdat verder de simultane verdeling van V symmetrisch is in X, Y en Z, is het gemakkelijk in te zien dat ook moet gelden: $Y \sim Bin(2, p_2)$ en $Z \sim Bin(2, p_3)$. Uitgeschreven levert dit:

$$M(t_1) \sim Bin(2, F(t_1)), \quad M(t_2) - M(t_1) \sim Bin(2, F(t_1) - F(t_2)) \quad \text{en} \quad 2 - M(t_2) \sim Bin(2, 1 - F(t_2)).$$

Dit resultaat vinden we trouwens ook wanneer we op de wat meer gebruikelijke manier de marginale verdelingen uit de simultane bepalen: door uitsommeren van de overige variabelen.

We merken ter volledigheid nog even op dat uiteraard ook $M(t_2) \sim Bin(2, F(t_2))$.

We zoeken nu de conditionele kans $P(M(t_2) = m_2 \mid M(t_1) = m_1) = P(X + Y = m_2 \mid X = m_1)$. We doen dit door deze kans voor alle mogelijke waarden van m_1 en m_2 uit te rekenen:

$$P(X + Y = 0 \mid X = 0) = \frac{P(\{X = 0\} \cap \{Y = 0\})}{P(X = 0)} = \frac{P(V = (0, 0, 2))}{(1 - p_1)^2} = \left(\frac{p_3}{1 - p_1}\right)^2. \text{ Zo ook:}$$

$$P(X + Y = 1 \mid X = 0) = 2\frac{p_2}{1 - p_1}\frac{p_3}{1 - p_1} = \frac{2p_2p_3}{(1 - p_1)^2},$$

$$P(X + Y = 2 \mid X = 0) = \left(\frac{p_2}{1 - p_1}\right)^2.$$

Merk op dat $1 - p_2/(1 - p_1) = p_3/(1 - p_1)$, dus: $(X + Y \mid X = 0) \sim Bin\big(2, p_2/(1 - p_1)\big)$.

Verder:

$$P(X + Y = 1 \mid X = 1) = \frac{p_3}{1 - p_1}$$

$$P(X + Y = 2 \mid X = 1) = \frac{p_2}{1 - p_1}$$

Dus $(X + Y \mid X = 1)$ heeft een alternatieve verdeling op $\{1, 2\}$ met parameter $p_2/(1 - p_1)$.

Tenslotte:

$$P(X + Y \mid X = 2) = 1.$$

Dus $(X + Y \mid X = 2)$ heeft een ontaarde verdeling in 2.

Dit alles leidt tot de conclusie dat de stochast $(X + Y \mid X = m_1)$ een naar m_1 verschoven $Bin\big(2 - m_1, p_2/(1 - p_1)\big)$-verdeling heeft.

3. Een meer realistisch model

Het in de vorige paragraaf beschreven model is niet erg realistisch, omdat we hebben aangenomen dat er twee fouten in de software zitten. In werkelijkheid zal dit aantal onbekend zijn.

We noemen vanaf nu het aantal fouten in de software u_0. Omdat de waarde hiervan onbekend is, is het een parameter in ons model. De waarde van deze parameter zullen we proberen te schatten.

Eén methode om dit te doen, is door kunstmatig een aantal fouten in de code te introduceren. Dit aantal noemen we M_0. Men gaat er vanuit dat al deze fouten even moeilijk te detecteren zijn als de u_0 fouten die al in de software aanwezig zijn. Vervolgens gaat men het programma testen, en vindt men N_1 fouten. Van deze N_1 fouten zaten er u_1 oorspronkelijk in de software en zijn er M_1 kunstmatig geïntroduceerd. Dus $N_1 = u_1 + M_1$.

Het ligt voor de hand om aan te nemen dat de verhouding $u_0 : M_0$ ongeveer gelijk zal zijn aan de verhouding $u_1 : M_1$. Daarom lijkt $\hat{u}_0 = u_1 M_0 / M_1$ een goede schatting voor u_0.

"lijkt" staat er, want in de praktijk blijken kunstmatige fouten makkelijker te ontdekken dan de fouten die oorspronkelijk in de software zitten. Daarom zal de schatting $\hat{u}_0$ te laag uitvallen als we de bovenstaande methode toepassen. Deze methode – die *fault seeding* heet – wordt dan ook niet vaak gebruikt.

We zullen nu ons model aanpassen aan de nieuwe situatie van een onbekend aantal fouten. We gaan er vanuit dat op tijdstip $t = 0$ u_0 fouten in de software aanwezig zijn. Deze fouten nummeren van 1 tot en met u_0. Aangenomen mag worden dat elke fout vroeg of laat tot een storing zal leiden, net als bij de situatie met 2 fouten. Conform de notatie uit de vorige paragraaf noemen we de tijd die verstrijkt tot fout nr. i tot een storing leidt T_i'. We nemen aan dat de stochasten $T_1', \ldots, T_{u_0}'$ onafhankelijk en gelijk verdeeld zijn, met verdelingsfunctie F en dichtheidsfunctie f. Ook wat betreft de notatie voor het tijdstip van de i-de storing houden we ons aan de notatie van de vorige paragraaf: T_i. Er geldt:

$$
\begin{aligned}
T_1 &= \min\{T_1', \ldots, T_{u_0}'\}, \\
T_2 &= \min\nolimits_1\{T_1', \ldots, T_{u_0}'\}, \\
T_3 &= \min\nolimits_2\{T_1', \ldots, T_{u_0}'\}, \\
&\vdots \\
T_{u_0} &= \max\{T_1', \ldots, T_{u_0}'\},
\end{aligned}
$$

waarbij $\min_i$ staat voor "de op i na kleinste".

Het aantal storingen voor of op tijdstip t noemen we weer $M(t)$. Merk op dat $M(t)$ nog steeds voldoet aan de eigenschap:

$$M(t) \geq i \quad \Leftrightarrow \quad T_i \leq t$$

Nu echter niet alleen voor $i \in \{1, 2\}$, maar voor alle $i \in \{1, \ldots, u_0\}$.

Zij $t_1, \ldots, t_m$ willekeurige tijdstippen, zodanig dat $0 < t_1 < \ldots < t_m$. Deze m tijdstippen verdelen de positieve reële rechte in $m+1$ disjuncte intervallen, namelijk $(0, t_1]$, $(t_1, t_2]$, $\ldots$, $(t_{m-1}, t_m]$ en (t_m, ∞).

Wat is nu de verdeling van $\big(M(t_1), M(t_2) - M(t_1), \ldots, M(t_m) - M(t_{m-1}), u_0 - M(t_m)\big)$: de vector die het aantal fouten in de afzonderlijke intervallen weergeeft? Het blijkt dat deze vector, net als in het geval waarbij $u_0 = 2$, een multinomiale verdeling heeft. Dit wordt duidelijk als we het optreden van de storingen in de intervallen weer bekijken als een serie onafhankelijke experimenten, waarbij elke mogelijke uitkomst (het 'terecht komen' van een storing in een bepaald interval) een vaste kans heeft om op te treden. Deze zienswijze is – ook nu u_0 onbekend is – gerechtvaardigd. We hebben immers aangenomen dat $T_1', \ldots, T_{u_0}'$ onafhankelijk en gelijk verdeeld zijn. Uit de eerder besproken theorie volgt nu dat het aantal storingen dat in de verschillende intervallen optreedt een multinomiale verdeling heeft. Rest ons nog de kansvector te bepalen. De kans dat de storing die hoort bij fout nr. i in een bepaald interval terecht komt, kan als volgt berekend worden:

$$\begin{aligned}
&P(T_i' \in (0, t_1]) = P(T_1' \leq t_1) = F(t_1), \\
&P(T_i' \in (t_1, t_2]) = P\big(\{T_1' \leq t_2\} \setminus \{T_1' \leq t_1\}\big) = F(t_2) - F(t_1), \\
&\qquad \vdots \\
&P(T_i' \in (t_{m-1}, t_m]) = P\big(\{T_1' \leq t_m\} \setminus \{T_1' \leq t_{m-1}\}\big) = F(t_m) - F(t_{m-1}), \\
&P(T_i' \in (t_m, \infty)) = P(T_1' > t_m) = 1 - P(T_1' < t_m) = 1 - F(t_m).
\end{aligned}$$

Dus $\big(M(t_1), M(t_2) - M(t_1), \ldots, M(t_m) - M(t_{m-1}), u_0 - M(t_m)\big)$ heeft een multinomiale verdeling met parameters u_0 en kansvector $\big(F(t_1), F(t_2) - F(t_1), \ldots, F(t_m) - F(t_{m-1}), 1 - F(t_m)\big)$.

4. Criteria voor de betrouwbaarheid

We zullen nu het begrip *storingsintensiteit* introduceren. Zoals de naam al doet vermoeden, geef dit de intensiteit weer waarmee storingen kunnen optreden. Hoe hoger de storingsintensiteit, hoe waarschijnlijker het is dat er in de nabije toekomst storingen zullen optreden. Uiteraard hangt de storingsintensiteit af van het aantal fouten dat in de software aanwezig is; hoe meer fouten, hoe hoger de storingsintensiteit. In de loop van de testperiode zal de intensiteit afnemen, omdat er steeds fouten worden gedetecteerd en verbeterd. Daarom is de storingsintensiteit afhankelijk van het tijdstip t. We zullen de storingsintensiteit in het vervolg aanduiden met de functie λ met parameter t. Voor hele kleine Δt geldt het volgende:

$$P(\text{storing in } (t, t + \Delta t)) \approx \lambda(t) \Delta t \,.$$

Ofwel:

$$\frac{P(\text{storing in } (t,\, t+\Delta t))}{\Delta t} \approx \lambda(t)\,.$$

Deze eigenschap kunnen we vertalen in een intuïtieve definitie van $\lambda(t)$:

$$\lambda(t) \coloneqq \lim_{\Delta t \to 0} \frac{P(\text{storing in } (t,\ t+\Delta t])}{\Delta t}\,.$$

Dit kunnen we verder uitwerken:

$$\lambda(t) \coloneqq \lim_{\Delta t \to 0} \frac{P(\text{storing in } (t,\ t+\Delta t])}{\Delta t} = \lim_{\Delta t \to 0} \frac{P(\exists i \in \{1,\ldots,u_0\} : T_i' \in (t, t+\Delta t])}{\Delta t} =$$
$$\lim_{\Delta t \to 0} \frac{P\left(\{T_1' \in (t, t+\Delta t]\} \cup \cdots \cup \{T_{u_0}' \in (t, t+\Delta t]\}\right)}{\Delta t} = \lim_{\Delta t \to 0} \frac{u_0 P(T_1' \in (t, t+\Delta t])}{\Delta t} =$$
$$\lim_{\Delta t \to 0} \frac{u_0 P(\{T_1' \le t+\Delta t\} \setminus \{T_1' \le t\})}{\Delta t} = \lim_{\Delta t \to 0} \frac{u_0 \left(F(t+\Delta t) - F(t)\right)}{\Delta t} =$$
$$u_0 \lim_{\Delta t \to 0} \frac{\left(F(t+\Delta t) - F(t)\right)}{\Delta t} = u_0 F'(t) = u_0 f(t).$$

Op een bepaald tijdstip t_e stopt men met testen. Als de storingsintensiteit op dat tijdstip laag genoeg is, is de software betrouwbaar genoeg om te worden vrijgegeven. Zo niet, dan zal men door moeten gaan met testen tot de software aan de gestelde eisen voldoet. Wat kunnen we eigenlijk zeggen over de storingsintensiteit op tijdstip t_e en de tijd daarna?

Als er na tijdstip t_e geen fouten meer worden gerepareerd, is de kans

$$P(\text{storing in } (t,\, t+\Delta t) \mid M(t_e) = m_e)$$

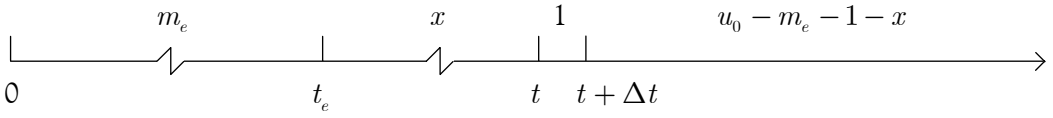
voor $t > t_e$ te bepalen.

Om te beginnen zullen we er vanuit gaan dat Δt dusdanig klein is, dat de kans op meer dan één storing binnen het interval $(t, t+\Delta t]$ verwaarloosbaar klein is. Dus:

$$P\left(\{\text{storing in } (t, t+\Delta t]\} \cap \{M(t_e) = m_e\}\right) =$$
$$P\left(\{M(t+\Delta t) - M(t) > 0\} \cap \{M(t_e) = m_e\}\right) \approx$$
$$P\left(\{M(t+\Delta t) - M(t) = 1\} \cap \{M(t_e) = m_e\}\right).$$

We noemen het aantal storingen dat binnen $(t_e, t]$ optreedt x. We kunnen de kans op één storing binnen het interval $(t,\ t+\Delta t]$ en m_e storingen binnen $(0, t_e]$ uitrekenen door te sommeren over de verschillende mogelijke waarden van x. Dat zijn $0, 1, 2, \ldots, u_0 - m_e - 1$, omdat er bij aanvang u_0 fouten in de software aanwezig zijn en elke fout uiteindelijk tot een storing leidt. Hieruit volgt bovendien dat het aantal fouten in het interval $(t+\Delta t,\ \infty)$ gelijk is aan $u_0 - m_e - 1 - x$.

Schematisch ziet de situatie er als volgt uit:



Definieer $V := \big(M(t_e), M(t) - M(t_e), M(t+\Delta t) - M(t), u_0 - M(t+\Delta t)\big)$. Van deze vector hebben we al laten zien dat hij een multinomiale verdeling heeft met parameters u_0 en kansvector $(F(t_e), F(t) - F(t_e), F(t+\Delta t) - F(t), 1 - F(t+\Delta t))$. Dit kunnen we als volgt gebruiken:

$$P\big(\{M(t+\Delta t) - M(t) = 1\} \cap \{M(t_e) = m_e\}\big) = \sum_{x=0}^{u_0 - m_e - 1} P\Big(V = (m_e,\, x,\, 1,\, u_0 - m_e - x - 1)\Big) =$$

$$\sum_{x=0}^{u_0 - m_e - 1} \binom{u_0}{m_e,\, x,\, 1,\, u_0 - m_e - 1 - x} F(t_e)^{m_e} \big(F(t) - F(t_e)\big)^x \big(F(t+\Delta t) - F(t)\big)\big(1 - F(t+\Delta t)\big)^{u_0 - m_e - 1 - x} =$$

$$\frac{u_0\,!}{m_e\,!} F(t_e)^{m_e} \big(F(t+\Delta t) - F(t)\big) \sum_{x=0}^{u_0 - m_e - 1} \frac{1}{x\,!(u_0 - m_e - x - 1)\,!} \big(F(t) - F(t_e)\big)^x \big(1 - F(t+\Delta t)\big)^{u_0 - m_e - 1 - x} =$$

$$\frac{u_0\,!}{m_e\,!(u_0 - m_e - 1)\,!} F(t_e)^{m_e} \big(F(t+\Delta t) - F(t)\big) \sum_{x=0}^{u_0 - m_e - 1} \binom{u_0 - m_e - 1}{x} \big(F(t) - F(t_e)\big)^x \big(1 - F(t+\Delta t)\big)^{u_0 - m_e - 1 - x}$$

$=: (*)$.

De som in $(*)$ is van het type:

$$\sum_{x=0}^{n} \binom{n}{x} a^x b^{n-x} = (a+b)^n.$$

Daarom vervolgen we de berekening als volgt:

$$(*) = (u_0 - m_e)\binom{u_0}{m_e} F(t_e)^{m_e} \big(F(t+\Delta t) - F(t)\big)\big(F(t) - F(t_e) + 1 - F(t+\Delta t)\big)^{u_0 - m_e - 1} =$$

$$(u_0 - m_e)\binom{u_0}{m_e} F(t_e)^{m_e} \frac{F(t+\Delta t) - F(t)}{\Delta t} \big(1 - F(t_e) + F(t) - F(t+\Delta t)\big)^{u_0 - m_e - 1} \Delta t =: (**).$$

Er geldt:

$$\frac{F(t+\Delta t) - F(t)}{\Delta t} \approx f(t).$$

Ook de afschatting $1 - F(t_e) + F(t) - F(t+\Delta t) \approx 1 - F(t_e)$ is op zijn plaats. Er volgt:

$$(**) \approx (u_0 - m_e)\binom{u_0}{m_e} F(t_e)^{m_e} f(t)\big(1-F(t_e)\big)^{u_0-m_e-1}\,\Delta t =$$

$$(u_0 - m_e)\frac{f(t)}{1-F(t_e)}\Delta t\binom{u_0}{m_e} F(t_e)^{m_e}\big(1-F(t_e)\big)^{u_0-m_e} =$$

$$(u_0 - m_e)\frac{f(t)}{1-F(t_e)}\Delta t\, P(M(t_e)=m_e).$$

Resumerend:

$$P\big(\{\text{storing in } (t,t+\Delta t]\}\cap\{M(t_e)=m_e\}\big) \approx (u_0 - m_e)\frac{f(t)}{1-F(t_e)}\Delta t\, P(M(t_e)=m_e).$$

Nu volgt heel eenvoudig:

$$P(\text{storing in } (t,\ t+\Delta t]\ \mid\ M(t_e)=m_e) = \frac{P\big(\{\text{storing in } (t,t+\Delta t]\}\cap\{M(t_e)=m_e\}\big)}{P(M(t_e)=m_e)} \approx$$

$$(u_0 - m_e)\frac{f(t)}{1-F(t_e)}\Delta t\, P(M(t_e)=m_e)\Big/ P(M(t_e)=m_e) = (u_0 - m_e)\frac{f(t)}{1-F(t_e)}\Delta t\,.$$

De storingsintensiteit die behoort bij de voorwaardelijke kans

$$P(\text{storing in } (t,\ t+\Delta t)\ \mid\ M(t_e)=m_e)$$

zullen we de *voorwaardelijke storingsintensiteit* noemen, en is gelijk aan:

$$\lambda(t\,|\,M(t_e)=m_e) = \frac{P(\text{storing in } (t,\ t+\Delta t]\ \mid\ M(t_e)=m_e)}{\Delta t} = (u_0 - m_e)\frac{f(t)}{1-F(t_e)}.$$

Bij het bepalen van de betrouwbaarheid van de software op een bepaald moment kunnen we deze voorwaardelijke storingsintensiteit goed gebruiken. Het aantal storingen dat optreedt in een kort interval $(t_1,t_2]$ is namelijk bij benadering Poisson verdeeld met parameter $\int_{t_1}^{t_2}\lambda(s)\,ds$. Dit is als volgt in te zien.

Stel dat $(t_1,\ t_2]$ een kort interval is. Op bepaalde momenten tijdens het testen vinden storingen plaats. Op kleine intervallen is de storingsintensiteit vrijwel constant. We kunnen het proces op zo'n interval dan opvatten als een Poissonproces. Het aantal storingen in $(t_1,t_2]$ is dus bij benadering Poisson verdeeld. Nu de parameter: deze moet gelijk zijn aan het verwachtte aantal storingen in $(t_1,\ t_2]$. Dit is precies het aantal fouten in de software maal de kans dat voor een gegeven fout de bijbehorende storing optreedt in $(t_1,\ t_2]$, dus $u_0P(T_1'\in(t_1,t_2])$. Dit uitwerken levert:

$$u_0P(T_1'\in(t_1,t_2]) = u_0\big(F(t_2)-F(t_1)\big) = u_0\int_{t_1}^{t_2} f(s)\,ds = \int_{t_1}^{t_2} u_0 f(s)\,ds = \int_{t_1}^{t_2}\lambda(s)\,ds\,.$$

Conclusie: $M(t_2)-M(t_1) \sim Pois(\int_{t_1}^{t_2}\lambda(s)\,ds)$ (bij benadering).

Tenslotte kijken we even naar een bijzonder geval: het interval $(0,\ t]$. Het verwachte aantal storingen $\mu(t)$ in dit interval is:

$$\mu(t) = E(M(t)) = \int_0^t \lambda(s)\,ds = u_0\left(F(t) - F(0)\right) = u_0 F(t).$$

Merk op dat we in de paragraaf 1 al voor het geval $u_0 = 2$ hadden opgemerkt dat $E(M(t)) = 2F(t)$.

We zullen vanaf nu de eis dat het interval kort moet zijn loslaten en ervan uitgaan dat voor ieder interval $(t_1,\ t_2]$ het aantal storingen dat erin optreedt Poisson verdeeld is met parameter $\int_{t_1}^{t_2} \lambda(s)\,ds$.

Stel nu dat we de volgende situatie hebben: een testteam heeft tot en met tijdstip t_e de software getest op fouten en vraagt zich af of de software nu betrouwbaar genoeg is. Het management heeft bepaald dat "betrouwbaar genoeg" wil zeggen: de kans dat er de eerstvolgende 4 CPU-uren geen fouten optreden, is tenminste 95%. We nemen aan dat er bij aanvang van het testen 100 fouten in de software zaten, en dat er daarvan 90 zijn gedetecteerd en gerepareerd. Er bevinden zich dus nog 10 fouten in de code. Bovendien zullen we aannemen dat $F(t) = 1 - e^{-10^{-6}t}$. T_i' heeft dus een exponentiële verdeling met parameter 10^{-6}. Er geldt dan: $f(t) = 10^{-6}e^{-10^{-6}t}$.

Met deze gegevens is de voorwaardelijke storingsintensiteit uit te rekenen. Aldus:

$$\lambda(t \mid M(t_e) = m_e) = (u_0 - m_e)\frac{f(t)}{1 - F(t_e)} = (100 - 90)\frac{10^{-6}e^{-10^{-6}t}}{1 - (1 - e^{-10^{-6}t_e})} = 10^{-5}e^{-10^{-6}(t - t_e)}.$$

Het aantal storingen dat de eerste 4 CPU-uren na het testen op zal treden kunnen we in wiskundige bewoordingen vertalen als het aantal storingen binnen het interval $(t_e,\ t_e + 4\cdot 60\cdot 60] = (t_e,\ t_e + 14400]$. Dit aantal heeft Poissonverdeling met parameter $\int_{t_e}^{t_e+14400} \lambda(s \mid M(t_e) = m_e)\,ds$. Nu de voorwaardelijke storingsintensiteit bekend is, is de parameter expliciet te bepalen.

$$\int_{t_e}^{t_e+14400} \lambda(s \mid M(t_e) = m_e)\,ds = 10^{-5}\int_{t_e}^{t_e+14400} e^{-10^{-6}(s-t_e)}ds \overset{(s^* = s - t_e)}{=} 10^{-5}\int_0^{14400} e^{-10^{-6}s^*}ds^* =$$

$$\frac{10^{-5}}{-10^{-6}}\left[e^{-10^{-6}s^*}\right]_{s^*=0}^{14400} = -10(e^{-\frac{9}{625}} - 1) = -10e^{-\frac{9}{625}} + 10 =: \theta.$$

De kans op geen enkele storing binnen het interval $(t_e,\ t_e + 14400]$ is dan:

$$P(\text{geen storing in } (t_e, t_e + 14400]) = \theta^0\,\frac{e^{-\theta}}{0!} = e^{10e^{-9/625} - 10} \approx 0,867.$$

De kans dat er geen storing optreedt in de eerste 4 CPU-uur na het testen is dus circa 86,7%. Dat is volgens het management niet genoeg om de software vrij te kunnen geven. Op tijdstip t_e is de software dus nog niet betrouwbaar genoeg.

Het management kan zich nu de vraag stellen hoe lang nog moet worden doorgegaan met testen totdat de software betrouwbaar genoeg is. Als men immers langer doorgaat met testen, zullen er meer

storingen optreden en zodoende meer fouten worden ontdekt en verbeterd. Daardoor neemt de storingsintensiteit af.

Laten we zeggen dat er na tijdstip t_e nog $\tilde{t}$ CPU-seconden wordt doorgegaan met testen. De kans dat er de eerste 4 CPU-uur na die tijd (dus na $t_e + \tilde{t}$ CPU-seconden) geen storingen optreden, is dan:

$$\int_{t_e+\tilde{t}}^{t_e+\tilde{t}+14400} \lambda(s \mid M(t_e) = m_e)\, ds = 10^{-5} \int_{t_e+\tilde{t}}^{t_e+\tilde{t}+14400} e^{-10^{-6}(s-t_e)} ds \overset{(s^*=s-t_e)}{=} 10^{-5} \int_{\tilde{t}}^{\tilde{t}+14400} e^{-10^{-6}s^*} ds^* =$$

$$\frac{10^{-5}}{-10^{-6}} \left[e^{-10^{-6}s^*}\right]_{s^*=\tilde{t}}^{\tilde{t}+14400} = -10(e^{-10^{-6}(\tilde{t}+14400)} - e^{10^{-6}\tilde{t}}) = -10e^{-10^{-6}\tilde{t}}(e^{-\frac{9}{625}} - 1) =: \theta.$$

En dus:

$$P(\text{geen storing in } (t_e + \tilde{t}, t_e + \tilde{t} + 14400)) = \theta^0 \frac{e^{-\theta}}{0!} = e^{10e^{-10^{-6}\tilde{t}}(e^{-9/625}-1)}.$$

Het met behulp van de computer gelijk stellen van deze kans met 0,95 levert op:

$$\tilde{t} \approx 1\,025\,062\,.$$

Met andere woorden: men zal nog ca. $1\,025\,062$ CPU-seconden ($\approx 11,9$ CPU-dagen) door moeten gaan met testen voordat de software voldoende betrouwbaar is.

In het laatste deel van deze paragraaf hebben we het onszelf wel erg makkelijk gemaakt door de waarde van u_0 en de verdeling $F(t)$ bekend te veronderstellen. In werkelijkheid zullen deze onbekend zijn, en moeten daarom geschat worden. Dat is het onderwerp van de volgende paragraaf.

5. Meest aannemelijke schatters

We nemen vanaf nu aan dat voor alle u_0 fouten het tijdstip waarop de bijbehorende storing optreedt dezelfde verdeling heeft, en dat de verdelingsfunctie gegeven wordt door $F_\beta(t) = 1 - e^{-\beta t}$. Met andere woorden: $T_i' \sim Exp(\beta)$ voor alle $i \in \{1, \ldots, u_0\}$. We gaan nu proberen schatters te vinden voor de twee parameters in ons model: u_0 en β, en gebruiken daarvoor meest aannemelijke schatters.

De meest aannemelijke schatters van u_0 en β zijn die waarden $\hat{u}_0$ en $\hat{\beta}$ die de kans op het waarnemen van de feitelijke realisaties zo groot mogelijk maken. We hebben getest over een vaste periode $(0, t_e]$ en we namen m_e storingen waar op de tijdstippen $t_1, \ldots, t_m$. Dit aantal storingen was van tevoren onbekend, dus dit getal moeten we zien als een of andere realisatie. Het is het grootste getal m_e met $t_{m_e} \leq t_e$ en $t_{m_e+1} > t_e$. Dus de kans op het waarnemen van de feitelijke realisaties zou gegeven moeten worden door:

$$P(T_1 = t_1, \ldots, T_{m_e} = t_{m_e}, T_{m_e+1} > t_e)\,.$$

Maar deze kans is gelijk aan 0, daar de verdeling van T_i continu is. Hier valt dus niets te maximaliseren. We zullen daarom kijken naar de kans:

$$p_{u_0,\beta} := P\left(T_1 \in \left(t_1 - \frac{\Delta t}{2},\, t_1 + \frac{\Delta t}{2}\right], \ldots,\, T_{m_e} \in \left(t_{m_e} - \frac{\Delta t}{2},\, t_{m_e} + \frac{\Delta t}{2}\right],\, T_{m_e+1} > t_e\right)$$

met Δt heel klein.

We betrekken hierbij de stochast $M(t)$ uit paragraaf 2 en 3 en vinden:

$$\begin{aligned} p_{u_0,\beta} = \; & P\left(M\left(t_1 - \frac{\Delta t}{2}\right) = 0,\; M\left(t_1 + \frac{\Delta t}{2}\right) - M\left(t_1 - \frac{\Delta t}{2}\right) = 1,\right. \\ & \left. \ldots, M(t_e) - M\left(t_e - \frac{\Delta t}{2}\right) = 0,\; u_0 - M(t_e) = u_0 - m_e\right). \end{aligned}$$

We kunnen hiervoor de in paragraaf 3 gevonden multinomiale verdeling gebruiken. Er volgt:

$$\begin{aligned} p_{u_0,\beta} = \; & \frac{u_0!}{0!1!\cdots 0!(u_0 - m_e)!}\left(F_\beta\left(t_1 - \frac{\Delta t}{2}\right)\right)^0 \prod_{i=1}^{m_e-1}\left(F_\beta\left(t_{i+1} - \frac{\Delta t}{2}\right) - F_\beta\left(t_i + \frac{\Delta t}{2}\right)\right)^0 \\ & \left(F_\beta(t_e) - F_\beta\left(t_{m_e} + \frac{\Delta t}{2}\right)\right)^0 \prod_{i=1}^{m_e}\left(F_\beta\left(t_i + \frac{\Delta t}{2}\right) - F_\beta\left(t_i - \frac{\Delta t}{2}\right)\right)^1 \left(1 - F_\beta(t_e)\right)^{u_0 - m_e} = \\ & u_0(u_0 - 1)\cdots(u_0 - m_e + 1)\prod_{i=1}^{m_e}\left(\frac{F_\beta(t_i + \frac{\Delta t}{2}) - F_\beta(t_i - \frac{\Delta t}{2})}{\Delta t}\Delta t\right)\left(1 - F_\beta(t_e)\right)^{u_0 - m_e} \approx \\ & \prod_{i=1}^{m_e}\left((u_0 - i + 1)\frac{F_\beta(t_i + \Delta t) - F_\beta(t_i)}{\Delta t}\Delta t\right)\left(1 - F_\beta(t_e)\right)^{u_0 - m_e} \approx \\ & \prod_{i=1}^{m_e}\left[(u_0 - i + 1) f_\beta(t_i)\Delta t\right]\left(1 - F_\beta(t_e)\right)^{u_0 - m_e} \end{aligned}$$

Als we hieruit alle Δt's wegdelen krijgen we de likelihood-functie:

$$L(\beta, u_0) = \prod_{i=1}^{m_e}\left[(u_0 - i + 1) f_\beta(t_i)\right]\left(1 - F_\beta(t_e)\right)^{u_0 - m_e} = \prod_{i=1}^{m_e}\left[(u_0 - i + 1)\beta e^{-\beta t_i}\right](e^{-\beta t_e})^{u_0 - m_e}$$

Zoals gebruikelijk nemen we hiervan de logaritme en we krijgen de log-likelihood:

$$\ell(\beta, u_0) = \sum_{i=1}^{m_e}\left[\log(u_0 - i + 1) + \log\beta - \beta t_i\right] - \beta t_e(u_0 - m_e)$$

Om nu $\hat{\beta}$ te bepalen, differentiëren we de log-likelihood naar β en stellen we de afgeleide gelijk aan 0.

$$\frac{\partial}{\partial\beta}\ell(\beta, u_0) = \sum_{i=1}^{m_e}\left[\frac{1}{\beta} - t_i\right] - t_e(u_0 - m_e) = \frac{m_e}{\beta} - \sum_{i=1}^{m_e} t_i - t_e(u_0 - m_e) = 0.$$

We vinden nu $\hat{\beta}$ door β hieruit op te lossen en voor u_0 de nog te bepalen $\hat{u}_0$ te substitueren.

$$\hat{\beta} = \frac{m_e}{\sum_{i=1}^{m_e} t_i + t_e(\hat{u}_0 - m_e)}.$$

Op dezelfde manier differentiëren we de log-likelihood naar u_0 en stellen die gelijk aan 0. We vullen voor β de gevonden $\hat{\beta}$ in:

$$\frac{\partial}{\partial u_0}\ell(\beta, u_0) = \sum_{i=1}^{m_e}\left[\frac{1}{u_0 - i + 1}\right] - \hat{\beta}t_e = \sum_{i=1}^{m_e}\frac{1}{u_0 - i + 1} - \frac{m_e t_e}{\sum_{i=1}^{m_e} t_i + t_e(u_0 - m_e)}$$
$$= 0 \Leftrightarrow \sum_{i=1}^{m_e}\frac{1}{\hat{u}_0 - i + 1} = \frac{m_e t_e}{\sum_{i=1}^{m_e} t_i + t_e(\hat{u}_0 - m_e)}$$

Dus $\hat{u}_0$ is de oplossing van:

$$\sum_{i=1}^{m_e}\frac{1}{\hat{u}_0 - i + 1} = \frac{m_e t_e}{\sum_{i=1}^{m_e} t_i + t_e(\hat{u}_0 - m_e)}.$$

In de praktijk zal deze oplossing benaderd moeten worden.

6. De testresultaten voor StatWorks

Het testteam heeft gedurende 91208 CPU-seconden het pakket StatWorks getest en de tijdstippen geregistreerd waarop een storing optrad. Deze tijdstippen kunt u terugvinden in de tabel van de bijlage.

We hebben bepaald dat het volgende geldt:

- $m_e = 136$.
- $t_e = 91208$.
- $\sum_{i=1}^{m_e} t_i = 3\,365\,957$.

Voor het bepalen van de meest aannemelijke schatting van u_0 moeten we nu oplossen:

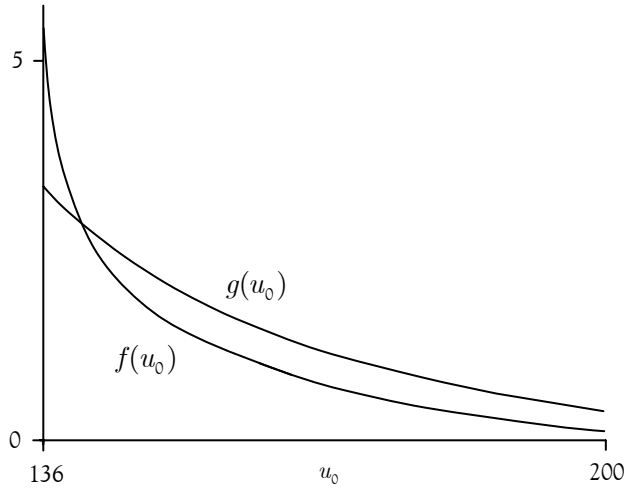
$$\sum_{i=1}^{136}\frac{1}{u_0 - i - 1} = \frac{136 \cdot 91208}{3\,365\,957 + 91208(u_0 - 136)}.$$

Heel belangrijk om op te merken is dat $u_0 \geq m_e = 136$. Het kan immers nooit zo zijn dat het aantal fouten dat op tijdstip $t = 0$ in de code aanwezig was kleiner is dan het aantal ontdekte fouten. We zijn er immers vanuit gegaan dat tijdens het reparatieproces geen nieuwe fouten optreden. Deze opmerking is van cruciaal belang, want er zijn heel veel oplossingen van de bovenstaande vergelijking die niet aan de deze eis voldoen.

Met behulp van Maple hebben we $\hat{u}_0$ bepaald door gebruik te maken van de volgende code:

```
f := u_0 -> sum(1/(u_0-i+1), i=1..136);
g := u_0 -> (136*91208)/(3365957+91208*(u_0-136));
fsolve(f(u_0)=g(u_0), u_0, 136..infinity);
```

De uitkomst die wij kregen, was $\hat{u}_0 \approx 141,01$. Dat dit ook werkelijk de enige oplossing is van $f(u_0) = g(u_0)$ (met f en g als in de bovenstaande Maple-code), volgt uit het verloop van beide functies.



Het vinden van een schatting voor β is nu slechts een kwestie van invullen.

$$\hat{\beta} = \frac{136}{3\,365\,957 + 91\,208\,(141,01 - 136)} \approx 3,56 \cdot 10^{-5}.$$

Het management vindt de software betrouwbaar genoeg als de kans dat de software de eerste 4 CPU-uur na het vrijgeven foutloos draait tenminste 95% is. Op dezelfde manier als in paragraaf 4 kunnen we nu bepalen hoe lang er nog moet worden doorgegaan met testen voordat de software kan worden vrijgegeven. Hiertoe bepalen we eerst de voorwaardelijke storingsintensiteit:

$$\begin{aligned}\lambda(t \,|\, M(92108) = 136) &= (\hat{u}_0 - 136)\frac{f(t)}{1 - F(92108)} \approx (141,01 - 136)\frac{3,56 \cdot 10^{-5} e^{-3,56 \cdot 10^{-5} t}}{1 - (1 - e^{-3,56 \cdot 10^{-5} \cdot 92108})} \\ &= 5,01 \cdot 3,56 \cdot 10^{-5} e^{-3,56 \cdot 10^{-5}(t - 92108)}.\end{aligned}$$

Vervolgens merken we op dat het aantal storingen dat optreedt in het interval $(t_e + \tilde{t},\ t_e + \tilde{t} + 14400]$ bij benadering Poisson verdeeld is met parameter $\int_{92108+\tilde{t}}^{92108+\tilde{t}+14400} \lambda(s \mid M(92108) = 136)\, ds$, waarbij:

$$\int_{92108+\tilde{t}}^{92108+\tilde{t}+14400} \lambda(s \mid M(92108) = 136)\, ds = 5,01 \cdot 3,56 \cdot 10^{-5} \int_{92108+\tilde{t}}^{92108+\tilde{t}+14400} e^{-3,56 \cdot 10^{-5}(s - 92108)}\, ds \overset{(s^* = s - 92108)}{=}$$

$$5,01 \cdot 3,56 \cdot 10^{-5} \int_{\tilde{t}}^{\tilde{t}+14400} e^{-3,56 \cdot 10^{-5} s^*}\, ds^* = -5,01 \left[e^{-3,56 \cdot 10^{-5} s^*}\right]_{s^* = \tilde{t}}^{\tilde{t}+14400} =$$

$$-5,01(e^{-3,56 \cdot 10^{-5}(\tilde{t}+14400)} - e^{-3,56 \cdot 10^{-5} \tilde{t}}) = -5,01(e^{-3,56 \cdot 10^{-5} \cdot 14400} - 1)e^{-3,56 \cdot 10^{-5} \tilde{t}} =: \theta.$$

Op de inmiddels bekende manier bepalen we de kans op geen enkele storing binnen $(t_e + \tilde{t},\ t_e + \tilde{t} + 14400]$.

$$P(\text{geen storing in } (t_e + \tilde{t},\ t_e + \tilde{t} + 14400]) = \theta^0 \frac{e^{-\theta}}{0!} = e^{5{,}01(e^{-3{,}56\cdot 10^{-5}\cdot 14400}-1)e^{-3{,}56\cdot 10^{-5}\tilde{t}}}.$$

De software is betrouwbaar genoeg als deze kans minimaal gelijk is aan 0,95. Oplossen van de vergelijking $e^{5{,}01(e^{-3{,}56\cdot 10^{-5}\cdot 14400}-1)e^{-3{,}56\cdot 10^{-5}\tilde{t}}} \geq 0{,}95$ levert (bij benadering) op:

$$\tilde{t} \geq 103\,070.$$

Men zal dus nog ca. 28,6 CPU-uren door moeten gaan met testen voordat de software voldoende betrouwbaar is.

7. Van CPU-tijd naar kalendertijd

De schatting die we in de vorige paragraaf hebben gemaakt, is een schatting in CPU-uren voor de resterende tijd voordat het programma kan worden vrijgegeven. Interessanter is het om een schatting te hebben in kalendertijd. Dan kan pas echt een verwachting worden gemaakt van de releasedatum van het programma. De vraag die we in deze paragraaf zullen proberen te beantwoorden, is hoe de CPU-tijd en werkelijke tijd met elkaar in verhouding staan.

De werkelijke tijd die aan de tests wordt besteedt, zal uiteraard meer zijn dan de CPU-tijd. Zo zal het testteam bijvoorbeeld gegevens moeten invoeren. Gedurende deze tijd is de computer niet aan het rekenen, dus telt deze tijd niet mee voor de CPU-tijd. Stel dat een CPU-uur voor het testen van de software in werkelijkheid 2½ mensuren kost. Als het testteam uit meerdere personen bestaat, kan het werk over de mensen worden verdeeld. Stel dat het testteam uit twee personen bestaat. Dan zijn beide leden van het team per CPU-uur 1¼ uur aan het werk. Het optreden van een storing kost het testteam extra tijd. Er zal dan namelijk een rapport moeten worden opgesteld van de storing, zodat het reparatieteam weet wat het met de storing aanmoet. Er zal bijvoorbeeld genoteerd moeten worden welke input de storing veroorzaakte, wat de aard van de storing was, enzovoorts. We nemen aan dat het opmaken van een rapport het testteam 1½ mensuur extra werk oplevert. In ons geval is dat dus ¾ uur per persoon, ofwel 2700 seconden.

Uit paragraaf weten we nog dat het verwachte aantal storingen dat in een bepaald tijdsinterval optreedt kan worden berekend met behulp van de storingintensiteit. We hadden beredeneerd dat het aantal storingen dat optreedt in een interval $(t_1, t_2]$ bij benadering Poisson verdeeld is met parameter $\int_{t_1}^{t_2} \lambda(s)\,ds$. Het verwachte aantal storingen dat in het interval optreedt, is dan gelijk aan de parameter. Zij τ de tijd gemeten vanaf tijdstip t_e. Is t de 'gewone' tijd (gemeten vanaf tijdstip 0), dan geldt dus $\tau = t - t_e$. Uiteindelijk zullen we voor τ de waarde van $\tilde{t}$ invullen die we aan eind van de vorige paragraaf hebben gevonden.

Het verwachte aantal storingen dat optreedt in het interval $(0,\tau]$ gegeven $M(t_e) = m_e$ zullen we aanduiden met $\tilde{\mu}(\tau)$. In analogie met $\mu(t)$ is $\tilde{\mu}(\tau)$ te schrijven als $\int_0^\tau \tilde{\lambda}(s)\,ds$, waarbij:

$$\tilde{\lambda}(t) = \lambda(t_e + t \mid M(t_e) = m_e) = (u_0 - m_e)\frac{f_\beta(t_e + t)}{1 - F_\beta(t_e)}.$$

Met behulp van de testresultaten en schattingen voor u_0 en β uit het begin van paragraaf 6 kunnen we een schatting geven van het aantal storingen dat optreedt in het tijdsinterval $(0,\tau]$. Dat gaat als volgt:

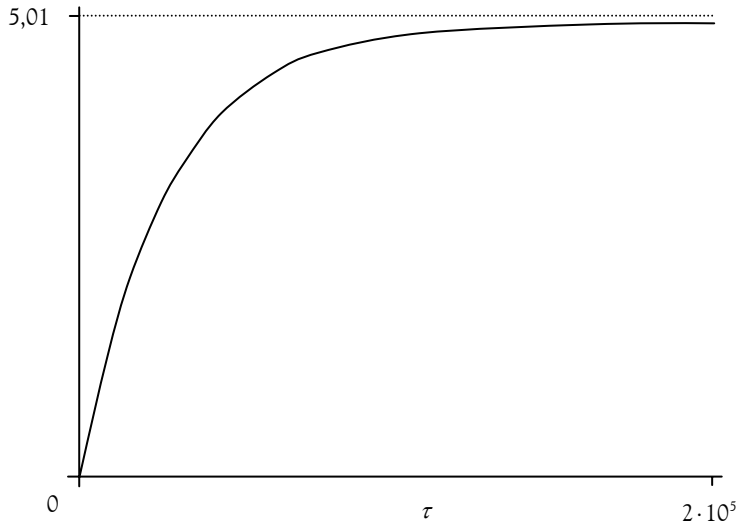
$$\tilde{\mu}(\tau) = \int_0^\tau \tilde{\lambda}(s)\,ds \overset{\text{verw.}}{=} \int_0^\tau (\hat{u}_0 - m_e)\frac{f_{\hat{\beta}}(t_e + s)}{1 - F_{\hat{\beta}}(t_e)}\,ds = \frac{(\hat{u}_0 - m_e)}{1 - F_{\hat{\beta}}(t_e)}\int_0^\tau f_{\hat{\beta}}(t_e + s)\,ds =$$

$$(\hat{u}_0 - m_e)e^{\hat{\beta}t_e}\int_0^\tau \hat{\beta}e^{-\hat{\beta}(t_e+s)}\,ds = -(\hat{u}_0 - m_e)e^{\hat{\beta}t_e}e^{-\hat{\beta}t_e}\left[e^{-\hat{\beta}s}\right]_{s=0}^{\tau} = (\hat{u}_0 - m_e)(1 - e^{-\hat{\beta}\tau}) =$$

$$= (141{,}01 - 136)(1 - e^{-3{,}56\cdot 10^{-5}\tau}) = 5{,}01(1 - e^{-3{,}56\cdot 10^{-5}\tau}).$$

Merk op dat deze uitdrukking nul is voor $\tau = 0$ en naar 5,01 convergeert als τ naar oneindig gaat, wat ook verwacht mag worden.

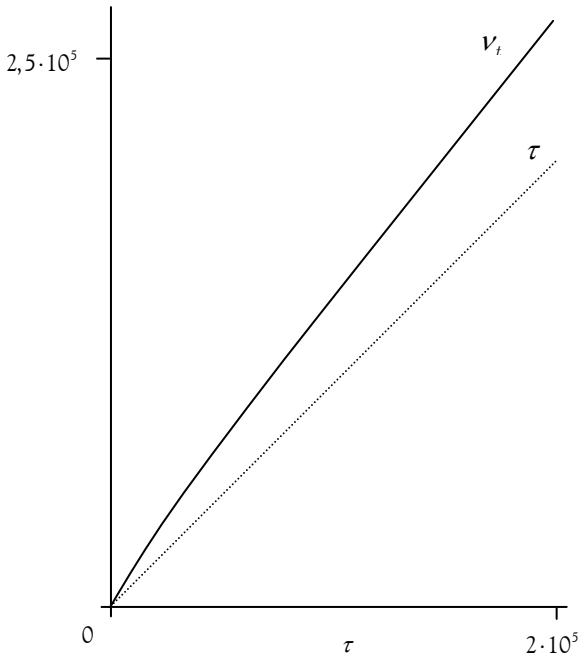
Hieronder ziet u de grafiek van $\tilde{\mu}(\tau)$ voor $\tau \in [0,\ 2\cdot 10^5]$.



Stel dat de leden van het testteam onbeperkt beschikbaar zijn. De verwachte kalendertijd (in seconden) die verstreken is na τ CPU-seconden noemen we $\nu_t(\tau)$. Uit het voorgaande is duidelijk dat geldt:

$$\nu_t(\tau) = 1{,}25\,\tau + 2700\tilde{\mu}(\tau).$$

De grafiek van $\nu_t(\tau)$ ziet er als volgt uit:

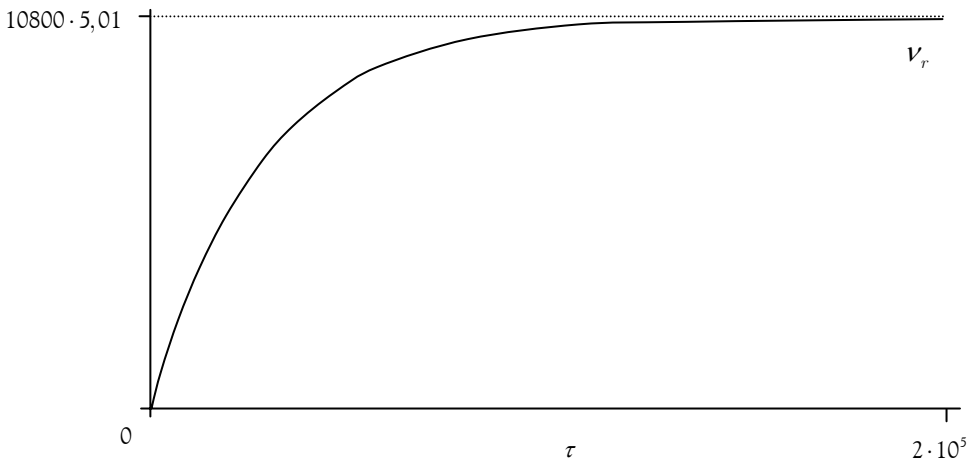


Naast het testteam is er een reparatieteam. De taak van dit team is het repareren van de fouten die het testteam heeft ontdekt. Het reparatieteam treedt dus alleen op als er een storing is opgetreden. Uiteraard zal het repareren van een fout meer tijd kosten dan alleen het constateren en rapporteren ervan. Daarom zal het reparatieteam per fout langer bezig zijn dan het teamteam. We nemen aan dat het repareren van een fout gemiddeld 6 mensuren in beslag neemt en dat het reparatieteam uit twee volledig inzetbare personen bestaat. Per persoon komt dit dus neer op drie uur per fout, ofwel 10800 seconden.

De verwachte kalendertijd die verstrijkt na τ CPU-seconden noteren we met $\nu_r(\tau)$. Er geldt:

$$\nu_r(\tau) = 10800\tilde{\mu}(\tau).$$

De grafiek van deze functie heeft het volgende verloop:

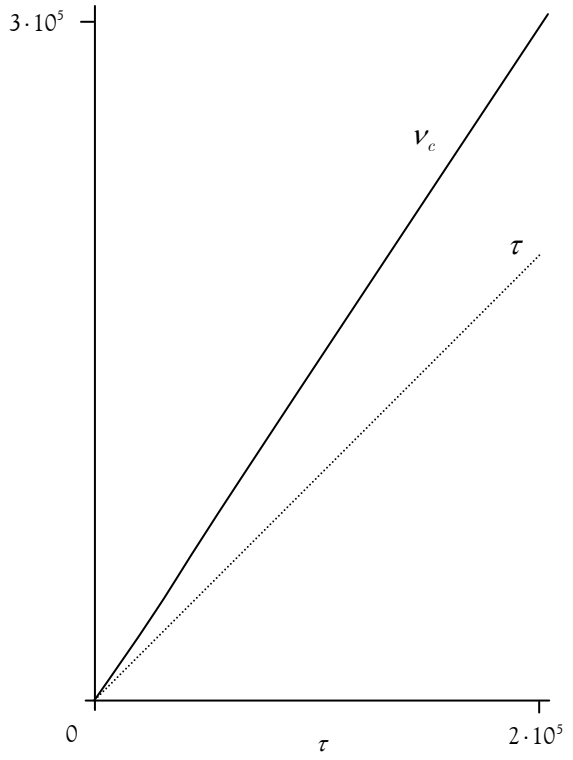


Tenslotte niet te vergeten de computerapparatuur. Laten we aannemen dat een uur CPU-tijd de computer 6 uur kost. Ook het optreden van een storing kost de computer tijd, omdat het testteam dan meestal het programma nog een aantal keer zal runnen om beter inzicht in te krijgen in de fout. We nemen aan dat elke opgetreden storing de computer een half uur kost.

In het bedrijf zijn vier computers beschikbaar. Per computer zal 1 CPU-seconde dus anderhalve seconde kosten. Een storing kost per computer 450 seconden extra. We noteren de verstreken kalendertijd die verstreken is na τ CPU-seconden als $\nu_c(\tau)$. Er geldt:

$$\nu_c(\tau) = 1,5\,\tau + 450\tilde{\mu}(\tau)$$

De grafiek van deze functie ziet er als volgt uit:

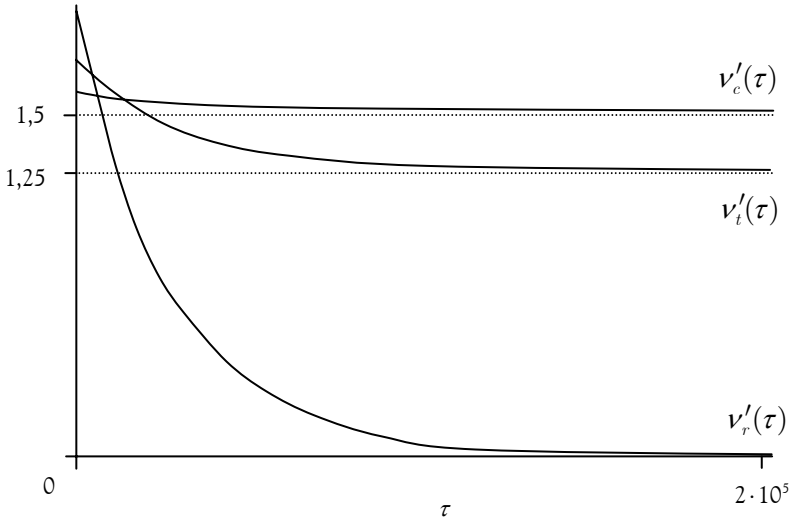


Gedurende het testen van de software wordt er tegelijkertijd een beroep gedaan op het testteam, het reparatieteam en de computerapparatuur. Meestal is aan het begin van de testprocedure het reparatieteam het zwaarst belast. Dat komt omdat er dan in een korte tijd relatief veel fouten ontdekt worden, omdat er nog veel fouten in de software aanwezig zijn. Het reparatieteam raakt dan zo zwaar belast, dat het testteam af en toe moeten wachten met testen. Na verloop van tijd zijn de meeste fouten gerepareerd en zullen er dus minder fouten gedetecteerd worden. Het reparatieteam heeft dan minder te doen, en wordt het testteam de zwaarst belaste groep. Als er nog langer wordt doorgegaan met testen, zal de beschikbare computertijd de beperkende factor worden.

Hoe bepalen we de beperkende factor op een bepaald tijdstip? Daarvoor moeten we kijken naar de bron die de grootste toename heeft van de kalendertijd, vergeleken met de computertijd. Deze kunnen we per bron berekenen door de functies ν_t, ν_c en ν_c te differentiëren. Dit gaat als volgt:

- $\nu_t'(\tau) = 1,25 + 2700\tilde{\mu}'(\tau) = 1{,}25+2700\tilde{\lambda}(\tau) = 1,25 + 2700(\hat{u}_0 - m_e)\dfrac{f_{\hat{\beta}}(t_e+\tau)}{1-F_{\hat{\beta}}(t_e)} =$
  $1,25 + 2700 \cdot 5{,}01 \cdot 3,56 \cdot 10^{-5} e^{-3{,}56\cdot 10^{-5}\tau} \approx 1,25 + 0,482e^{-3{,}56\cdot 10^{-5}\tau}$.
- $\nu_r'(\tau) = 10800\tilde{\mu}'(\tau) = 10800\tilde{\lambda}(\tau) = 10800 \cdot 5{,}01 \cdot 3,56 \cdot 10^{-5} e^{-3{,}56\cdot 10^{-5}\tau} \approx 1,926e^{-3{,}56\cdot 10^{-5}\tau}$.
- $\nu_c'(\tau) = 1,5 + 450\tilde{\mu}'(\tau) = 1,5 + 450\tilde{\lambda}(\tau) = 1,5 + 450 \cdot 5{,}01 \cdot 3,56 \cdot 10^{-5} e^{-3{,}56\cdot 10^{-5}\tau} \approx$
  $1,5 + 0,080e^{-3{,}56\cdot 10^{-5}\tau}$.

Grafisch ziet dit er als volgt uit:



Uit de grafiek is af te leiden dat de situatie bij MathWorks inderdaad voldoet aan de verwachting. Eerst is het reparatieteam de beperkende factor, daarna het testteam en tenslotte de computer-apparatuur.

Het ligt nu voor de hand om het uiteindelijke aantal verstreken seconden na τ CPU-seconden, dat we zullen noteren als $\nu(\tau)$, te definiëren zodat geldt:

$$\nu'(\tau) = \max\left\{\nu_t'(\tau), \nu_r'(\tau), \nu_c'(\tau)\right\}.$$

Het hangt immers steeds van de beperkende factor af hoe lang het testproces zal duren.

Om hieruit een expliciete uitdrukking voor $\nu(\tau)$ als te leiden, berekenen we allereerst de snijpunten van $\nu_t'(\tau)$, $\nu_r'(\tau)$ en $\nu_c'(\tau)$.

$$\begin{aligned}
\nu_r'(\tau) = \nu_c'(\tau) &\Leftrightarrow 10800\tilde{\lambda}(\tau) = 1,5 + 450\tilde{\lambda}(\tau) \\
&\Leftrightarrow \tilde{\lambda}(\tau) \approx 1,449 \cdot 10^{-4} \\
&\Leftrightarrow e^{-3,56\cdot 10^{-5}\tau} \approx 0,813 \\
&\Leftrightarrow \tau \approx 5830 =: \iota_1
\end{aligned}$$

$$\begin{aligned}
\nu_t'(\tau) = \nu_c'(\tau) &\Leftrightarrow 1,25 + 2700\tilde{\lambda}(\tau) = 1,5 + 450\tilde{\lambda}(\tau) \\
&\Leftrightarrow \tilde{\lambda}(\tau) \approx 1,111 \cdot 10^{-4} \\
&\Leftrightarrow e^{-3,56\cdot 10^{-5}\tau} \approx 0,623 \\
&\Leftrightarrow \tau \approx 13294 =: \iota_2
\end{aligned}$$

Uit het verloop van de grafieken van $\nu_t'(\tau)$, $\nu_r'(\tau)$ en $\nu_c'(\tau)$ volgt nu dat:

$$\nu'(\tau) = \begin{cases} \nu_r'(\tau) & \text{als} \quad \tau \in (0, \iota_1] \\ \nu_t'(\tau) & \text{als} \quad \tau \in (\iota_1, \iota_2] \\ \nu_c'(\tau) & \text{als} \quad \tau \in (\iota_2, \infty) \end{cases}$$

Er volgt:

$$\nu(\tau) = \begin{cases} \nu_r(\tau) & \text{als} \quad \tau \in (0, \iota_1] \\ \nu_t(\tau) + c_1 & \text{als} \quad \tau \in (\iota_1, \iota_2] \\ \nu_c(\tau) + c_2 & \text{als} \quad \tau \in (\iota_2, \infty) \end{cases}$$

Formeel gezien hadden we eigenlijk ook een constante achter $\nu_r(\tau)$ moeten zetten, maar het is direct duidelijk dat deze constante nul moet zijn.

Rest ons nog de waarde van c_1 en c_2 te bepalen. We gebruiken hiervoor de intuïtief duidelijke aanname dat ν een continue functie moet zijn. Er moet dan gelden:

$$\begin{aligned}
&\nu_r(\iota_1) = \nu_t(\iota_1) + c_1 \\
&10800\tilde{\mu}(\iota_1) = 1,25\,\iota_1 + 2700\tilde{\mu}(\iota_1) + c_1 \\
&c_1 = 8100\tilde{\mu}(\iota_1) - 1,25\,\iota_1 \\
&c_1 \approx 8100 \cdot 0,939 - 1,25 \cdot 5830 \approx 318,4.
\end{aligned}$$

$$\begin{aligned}
&\nu_t(\iota_2) + c_1 = \nu_c(\iota_2) + c_2 \\
&1,25\,\iota_2 + 2700\tilde{\mu}(\iota_2) + c_1 = 1,5\,\iota_2 + 450\tilde{\mu}(\iota_2) + c_2 \\
&c_2 = c_1 - 0,25\,\iota_1 + 2250\tilde{\mu}(\iota_2) \\
&c_2 \approx 318,4 - 0,25 \cdot 13294 + 2250 \cdot 1,889 \approx 1245,1.
\end{aligned}$$

De functie ν heeft dus (bij benadering) het volgende voorschrift:

$$\nu(\tau) = \begin{cases} \nu_r(\tau) & \text{als} \quad \tau \in (0, 5830] \\ \nu_t(\tau) + 318,4 & \text{als} \quad \tau \in (5830, 13294] \\ \nu_c(\tau) + 1245,1 & \text{als} \quad \tau \in (13294, \infty) \end{cases}$$

We hebben nu voldoende voorbereidingen getroffen om een echte schatting te kunnen maken van de releasedatum van het programma StatWorks. In paragraaf 6 hadden we geschat dat er nog circa 103 070 CPU-seconden voor het testen nodig zijn voordat het softwareprogramma kan worden vrijgegeven. Met de gevonden formule voor $\nu(\tau)$ kunnen we bepalen met welke kalendertijd dit overeenkomt. Aldus:

$$\nu(\tilde{t}) \approx \nu(103\,070) = \nu_c(103\,070) + 1245,1 =$$

$$1,5 \cdot 103\,070 + 450 \cdot 5,01 \cdot (1 - e^{-3,56 \cdot 10^{-5} \cdot 103\,070}) + 1245,1 \approx 158\,048.$$

De verwachting is dus dat de software over 156 802 seconden kan worden vrijgegeven. Dit komt overeen met 43,9 uur. Als we er vanuit gaan dat het personeel 40 uur per week werkt, kunnen we concluderen dat het naar verwachting nog iets meer dan een week duurt voordat het programma kan worden vrijgegeven.

Bijlage: Storingstijdstippen

3	1846	5324	10258	15806	26770	42296	56485
33	1872	5389	10491	16185	27753	42306	56560
146	1986	5565	10625	16229	28460	45406	57042
227	2311	5623	10982	16358	28493	46653	62551
342	2366	6080	11175	17168	29361	47596	62651
351	2608	6380	11411	17458	30085	48296	62661
353	2676	6477	11442	17758	32408	49171	63732
444	3098	6740	11811	18287	35338	49416	64103
556	3279	7192	12549	18568	36799	50145	64893
571	3288	7447	12559	18728	37642	52042	71043
709	4434	7644	12791	19556	37654	52489	74364
759	5034	7837	13121	20567	37915	52875	75409
836	5049	7843	13486	21012	39715	53321	76057
860	5085	7922	14708	21308	40580	53443	81542
968	5089	8738	15251	23063	42015	54433	82702
1056	5090	10089	15261	24127	42045	55381	84566
1726	5097	10237	15277	25910	42188	56463	88682