Modellering av trafikflöde

Andreas Källberg, andkal@student·chalmers·se
Robin Persson Söderholm, robper@student·chalmers·se
Simon Sigurdhsson, ssimon@student·chalmers·se

17 maj 2010

Sammanfattning

En makroskopisk modellering av trafikflöde i ett antal olika intressanta situationer, samt en kort analys av några av de metoder som finns att tillgå samt de resultat modellen har gett.

Innehåll

1 Introduktion

1.1 Problembeskrivning

Att modellera trafikflöde kan vara mycket viktigt i planeringen av infrastruktur. En modell som tar hänsyn till latens i trafiken (som uppstår pga reaktionstider hos förare och liknande) kan ge viktiga resultat som sedan kan tillämpas för att förbättra vägnätet och därmed förkorta restiden i nätet.

Tanken är att vi först modellerar några enklare trafiksituationer, till exmpel raka vägar (utan korsningar) med trafikljus, och sedan utökar dessa modeller för att slutgiltigen kunna sätta upp en större modell som innefattar korsningar med eller utan rödljus.

Genom att göra detta hoppas vi kunna identifiera mönster för hur man på ett lämpligt sätt synkroniserar trafikljus för att få ett bra trafikflöde, samt kunna identifiera några bra och dåliga sätt att bygga upp ett trafiknätverk.

Vi kommer at basera vår modell på en docentföreläsning (Diehl, 2006) av Stefan Diehl på LTH, främst genom att basera modellen på kontinuummekanik.

2 Analys

2.1 Lösning

Vi baserar vår modell på flödesmekanik genom att betrakta bilar som en vätska med täthet. Detta ger oss en makroskopisk modell av trafiken och dess flöde. Det är en naturlig generalisering att göra, vilket vi kommer att se senare. Vi börjar med att definiera ett antal olika stoheter: biltätheten $\rho(x,t)$, trafikfarten $v(x,t)$ och trafik*flödet* $q(x,t) = \rho v$.

Vidare antar vi att antalet bilar i systemet bevaras, dvs. att systemet lyder under kontinuitetsekvationen (1). Notera att indexnotationen indikerar att uttrycket skall deriveras med avseende på variabeln. För att förenkla saken ytterligare förutsätter vi till en början att vi har ett homogent trafikflöde, dvs. att trafikfarten v endast beror på ρ, enligt $v = V(\rho)$.

$$\rho_t + (\rho v)_x = 0 \tag{1}$$

Detta gör att vi kan införa en enklare beteckning för den andra termen i (1). Vi inför funktionen Q enligt $q = \rho v = \rho V(\rho) \equiv Q(\rho)$, som nu endast beror på ρ. Vi har alltså istället uttrycket i (2), vilket väsentligen är samma sak.

$$\rho_t + Q(\rho)_x = 0 \tag{2}$$

Vårt *trafikflöde* är nu Q istället för q, och beror helt och hållet på hur tät trafiken är. Vi förutsätter alltså att alla kör så fort som de får och kan med avseende på hur mycket trafik det är på vägarna. Vi kommer nu även att införa hastighetsbegränsningar i det här uttrycket. Vi kan även använda kedjeregeln för att få fram (3) och på så sätt se ett intressant faktum: $Q'(\rho)$ är egentligen en signalhastighet (jämför med c i vågekvationen), och beror på "amplituden" ρ.

$$\rho_t + Q'(\rho) \cdot \rho_x = 0 \tag{3}$$

Vi gör det rimliga antagandet att hastigheten avtar med ökad täthet, (dvs. $V'(\rho) \leq 0$) och ställer därför upp uttrycket (4). Detta ger oss möjlighet att till exempel variera maxhastigheten längs vägen (tänk hastighetsbegränsningar) och kanske till och med sänka denna till noll (tänk trafikljus). Skulle vi vilja det kan vi även variera den maximala tätheten, för att simulera ett förändrat antal filer på vägen eller dylikt.

$$V(\rho) = v_{max}\left(1 - \frac{\rho}{\rho_{max}}\right) \tag{4}$$

När vi ställt upp det enkla uttrycket i (4) kan vi enkelt ställa upp en ekvation som vi kan moddellera enkelt i `MATLAB`. Det finns många olika metoder att göra detta, bland annat kan vi använda `MATLAB`s inbyggda metoder för PDE-lösning (`pdepe`), eller dess metoder för ODE-lösning (t.ex. `ode45`) med en enkel numerisk derivering i ODE-funktionen (lämpligtvis med hjälp av `diff`). Vi skulle även kunna använda till exempel Lax-Wendroff. Denna diskuteras senare.

2.1.1 Kritiska gränsen

Av de ekvationer vi satt upp kan man se ett par intressanta saker; framför allt så kan man se att det finns en *kritisk gräns* för tätheten; beroende på om tätheten ligger över eller under denna gräns så kommer signalhastigheten ($Q'(\rho)$) att vara antingen positiv eller negativ. Vad detta i praktiken innebär är att köer "rör sig" bakåt när det är mycket trafik, medans de följer trafiken vid lite mindre trafiktäta situationer.

2.1.2 Hastighetsbegränsningar

Att nu införa diverse restriktioner på trafikflödet är enkelt. Vi kan införa hastighetsbegränsningar (och variationer av dessa) genom att helt enkelt se till att v_{max} beror på x, till exempel enligt (5). Det är enkelt att generalisera detta till att inkludera hur många hastighetsförändringar som helst.

$$v_{max}(x) = \begin{cases} 90 \text{ km/h}, & x \leq 5\text{km} \\ 50 \text{ km/h}, & x > 5\text{km} \end{cases} \tag{5}$$

2.1.3 Trafikljus

Genom att utnyttja v_{max} ännu mer kan man införa trafikljus. Man kan till exempel se till att variabeln beror både på x och t enligt något godtyckligt mönster (för ett trafikljus påverkas v_{max} lämpligen vid ett fixt x och sätts till noll vid periodiska tider t). Man kan även införa "adaptiva" trafikljus, som släpper igenom trafik om tätheten överskrider ett visst värde, och sedan slår om till rött igen om det sjunker under ett annat (förslagsvis lägre) värde. Det är dock svårt att göra i större utsträckning när man har korsningar (och därmed flera trafikflöden), och det blir dessutom svårt att implementera med den PDE-lösare vi valt att använda, därför har vi avstått från att göra detta.

Vi har dock valt att istället för att rakt av sätta v_{max} till noll i en given punkt skapa en slags "grop" med hjälp av en Gaussdistribution. Detta gör hastighetsförändringen mjukare och bidrar till att dämpa eventuella artefakter som introduceras av PDE-lösaren vi använder.

2.1.4 Sammanslagning av filer

Sammanslagning (eller delning) av filer emuleras enkelt genom att manipulera ρ_{max}. Tänk dig till exempel att två filer går ihop till en — detta kan man enkelt modellera genom att halvera ρ_{max}.

2.1.5 Korsningar

När vi satt upp kriterier för både trafikljus och sammanslagning av filer kan vi enkelt sätta upp modeller för några relativt enkla korsningar. Manipulerar vi modellen lite extra kan vi sammanfoga olika trafikflöden för att få en bild av hur trafiken påverkas av en korsning. Här följer ett par olika korsningar och hur man kan tänkas beskriva dessa i vår modell.

T-korsning/påfart; sammanslagning utan trafikljus
En T-korsning där två trafikflöden sammanfogas utan trafikljus är enkel att modellera. Man kan, i vårt högerled, manipulera ρ och addera en källtäthet vid en viss position och på så vis öka tätheten. Se till exempel (6) nedan, där s är denna tillagda täthet.

$$\rho_t + Q(\rho)_x = s \tag{6}$$

Man kan se detta fall antingen som en påfart, eller som en T-korsning där fler kör på från sidovägen än det kör av folk från huvudvägen.

T-korsning; sammanslagning med trafikljus
En variant av föregående korsning uppstår när man lägger till ett trafikljus. Detta är inte svårare att modellera, eftersom man inte får problemet med tätheten. Det man gör är helt enkelt att ställa upp en modell för en rak väg med trafikljus, och sedan när detta trafikljus är rött ser man till att det även genererar en täthet, som då representerar flödet som kommer från den andra vägen.

T-korsning; normal utan trafikljus
Ska man däremot modellera en korsning där man får svänga åt vilket håll man vill blir det lite svårare. Låt oss förutsätta att en bilist kommer att svänga av huvudvägen med en sannolikhet p. Vi måste då vid korsningen emulera detta genom att multiplicera tätheten med en faktor $(1-p)$. På detta måste vi sedan lägga en annan täthet som representerar den trafik från sidovägen som väljer att svänga in *i den riktning vi räknar på*. Detta blir som innan en källtäthet som adderas. Man kan även tolka mängden bilar som kör av som en negativ källtäthet. Man kan sedan superponera dessa två källtätheter för att få nettokälltätheten som ska läggas till (eller dras ifrån) vårt högerled. I övrigt får man samma problem som med den "vanliga" sammanslagningen, och dessa manifesterar sig på samma sätt.

T-korsning; normal med trafikljus
Vill vi utöka detta med trafikljus kombinerar vi helt enkelt föregående steg med trafikljus, precis som med den "vanliga" sammanslagningen. Skillnaden blir precis som innan en faktor $(1-p)$ (eller en nettokälltäthet), samt att trafikljuset nu endast ska generera den trafik som svänger ut *i rätt riktning*.

Fyrvägskorsning med trafikljus
En fyrvägskorsning är inte (modelleringsmässigt) olik en T-korsning på något sätt. Istället för att man nu endast har trafik från ena hållet så har man trafik från båda hållen. Man måste alltså justera p, och även justera det genererade flödet från sidovägarna. Lämpligtvis synkar man trafikljusen antingen så att båda kör samtidigt eller så att de två sidovägarna turas om.

Man skulle kunna tolka den här beskrivningen som två T-korsningar som man helt enkelt placerar ovanpå varandra. Det går alltså att tillämpa en viss superpositionsprincip i det här problemet.

Motorvägspåfart
En motorvägspåfart är helt enkelt en kombination av hastighetsförändring och sammanslagning. Hur man ska beskriva detta i vår modell beror främst på om man vill modellera flödet av bilar *på* motorvägen eller flödet av de på *påfarten*.

För flödet på motorvägen kan man anta att de som kommer från påfarten helt enkelt hinner accelerera *innan* påfarten är slut. Problemet reduceras då till en sammanfogning av filer. Modellerar man istället de som kommer från påfarten måste man göra en hastighetsökning till den hastighet trafiken på motorvägen håller och sedan göra en sammanslagning. Detta är inte nämnvärt svårare.

Man skulle även kunna utöka vår modell till två dimensioner och faktiskt ställa upp en påfart och flödet på båda grenar. Detta blir dock betydligt mer komplext och svårt att beräkna. Modellen i sig förändras bara i högerledet, där ρ_x istället blir $\rho_{\mathbb{x}}$. Hur man tolkar den derivatan (dvs är det $\nabla\rho$ eller $\nabla \cdot \rho$?) påverkar givetvis resultatet. Lämpligast är om man tolkar den som $\nabla\rho$, gradienten, och att Q (och därmed Q') beror på något annat än (4) — till exempel är det lämpligt att v_{max} beror på $\mathbb{x}$ och dessutom är en vektor, så att man kan "styra" flödet. Tätheten ρ_{max} kommer givetvis också bero på $\mathbb{x}$, men kommer fortfarande att vara skalär.

Vi kommer dock inte att utveckla vår modell så långt, även om en sådan utveckling klart skulle kunna förbättra modelleringen av många av våra problemställningar.

2.2 Metoder

Vi har utforskat en del metoder som kan användas för att lösa det system vi ställt upp tidigare. Dessa är metoder för att lösa *hyperboliska PDEer*, vilket är precis vad vi har. Lyckligtvis är detta ett område som är väl behandlat (då dessa ekvationer även dyker upp inom flödesmekaniken), och det finns därför en del intressanta metoder att titta på.

2.2.1 `ode45` & `diff`

Den enklaste metoden att ställa upp är en kombination av `ode45` och `diff` i `MATLAB`. I korthet går den ut på att man ställer upp ett ODE-system med avseende på den ena variabeln (lämpligtvis *tiden* i vårt fall) och approximerar derivatan med avseende på den andra variabeln med hjälp av `diff`. Det ger ett system som löses snabbt, men är relativ inexakt eftersom den numeriskt beräknade derivatan är relativt inexakt.

Själva beräkningarna utförs alltså till stor del av `ode45`, som använder en Runge-Kutta-metod för att lösa det ODE-problem som ställs upp. Derivatan ρ_x (eller $Q'(\rho)$, som vi valt att använda av stabilitetsskäl) beräknas med `diff`, som helt enkelt tar skillnaden mellan två närliggande element i en vektor. Dividerar vi då med steglängden så får vi en grov approximation av derivatan i våra punkter. Vi kommer att sakna en punkt, men det går att åtgärda.

2.2.2 Lax-Wendroff

Lax-Wendroff är en metod speciellt anpassad för att lösa hyperboliska PDEer. Den är mycket exakt, bortsett från att den introducerar artefakter i de fall där stora derivator förekommer. För att beskriva metoden betraktar vi vår PDE:

$$\frac{\partial \rho}{\partial t} = \frac{\partial Q(\rho)}{\partial x}$$

Metoden arbetar i två steg. Det första, Lax-steget (7), beräknar ρs värde i halvtids- och halvpositionspunkter. Nästa steg (8), tar fram värdet vid $t = n+1$ utifrån halvstegen.

$$\rho_{i+1/2}^{n+1/2} = \frac{\Delta t}{2\Delta x}\left(Q(\rho_i^n) - Q(\rho_{i+1}^n)\right) + \frac{\rho_i^n + \rho_{i+1}^n}{2} \tag{7}$$

$$\rho_i^{n+1} = \frac{\Delta t}{\Delta x}\left(Q(\rho_{i-1/2}^{n+1/2}) - Q(\rho_{i+1/2}^{n+1/2})\right) + \rho_i^n \tag{8}$$

2.2.3 MacCormack

MacCormack är en modifierad variant av Lax-Wendroff som är betydligt enklare att implementera. Modifikationen introducerar visserligen ringartefakter i de områden där derivatan är stor, men det är inget större problem i vår modell. Den är, liksom Lax-Wendroff, specifikt designad för att beräkna hyperboliska PDEer. För att beskriva metoden betraktar vi vår PDE:

$$\frac{\partial \rho}{\partial t} + \frac{\partial Q(\rho)}{\partial x} = 0$$

Metoden består av två steg; ett skattande steg och ett korrigerande steg. I det första steget beräknas ett preliminärt värde på u i nästa tidssteg, betecknat $\rho_i^{\overline{n+1}}$. Detta uppskattas enligt (9). I nästa steg korrigeras det uppskattade värdet med hjälp av (10).

$$\rho_i^{\overline{n+1}} = \rho_i^n - \frac{\Delta t}{\Delta x}\left(Q(\rho_{i+1}^n) - Q(\rho_i^n)\right) \tag{9}$$

$$\rho_i^{n+1} = \rho_i^{n+1/2} - \frac{\Delta t}{2\Delta x}\left(Q(\rho_i^{\overline{n+1}}) - Q(\rho_{i-1}^{\overline{n+1}})\right) \tag{10}$$

Nackdelen med detta andra steg är att man får halvtidssteg. Halvtidssteg är jobbiga att representera. Man brukar därför ersätta halvtidstermen i detta steg med *tidsgenomsnittet* (11), vilket gör metoden enklare att implementera.

$$\rho_i^{n+1/2} = \frac{\rho_i^n + \rho_i^{\overline{n+1}}}{2} \tag{11}$$

Man kan även om man vill alternera differentieringsordningen varje tidssteg (dvs. alternera mellan bakåt- och framåtdifferens i (9) och (10)) för att öka exaktheten. Metoden är extra lämplig för ickelinjära PDEer, och för linjära PDEer är metoden ekvivalent med Lax-Wendroff.

2.2.4 `pdepe`

Funktionen `pdepe` är `MATLAB`s inbyggda metod för att lösa (vissa typer av) PDE-system. Metoden löser relativt många PDEer, men är något svår att använda då man måste sätta upp många olika funktioner. Detta kan tyckas vara överdrivet arbetsintensivt för ett så "enkelt" problem som detta.

Metoden introducerar dock stora ringartefakter, som främst dyker upp när täthetssignalen passerar en diskontinuitetet, exempelvis ena flanken på en fyrkantsvåg. Ringartefakterna kan göra det svårt att tolka datan, varför vi valtt att köra utdatan från `pdepe` genom ett Gaussiskt filter för att dämpa dessa artefakter.

Ringartefakterna dyker främst upp när täthetssignalen passerar en diskontinuitetet, exempelvis ena flanken på en fyrkantsvåg.

Metoden opererar ungefär som kombinationen `ode45` och `diff`; den använder `ode15s` i tidsled och utnyttjar problemformuleringen för att lösa PDEen mycket effektivare (och exaktare) än den grova kombinationen av `ode45` och `diff`.

2.2.5 Gibbs fenomen (ringartefakter)

Alla de tre sista metoderna introducerar så kallade ringartefakter. Detta är på grund av Gibbs fenomen, och beror på att man vid diskontinuiteter över- och underskjuter det verkliga värdet när man beräknar derivatan. Det finns sätt att dämpa dessa artefakter, till exempel genom att som i MacCormack-metoden alternera mellan framåt- och bakåtderivata, eller genom att köra in- och utdata genom ett Gaussiskt filter. Detta har vi valt att göra i `pdepe`-metoden.

2.2.6 Gaussiska filter

För att bli av med ringartefakter kan man som sagt använda ett Gaussiskt filter. Ett sådant filter tar inte bort artefakter i sig utan verkar för att "mjuka upp" utdatan vi får. Man kan även köra ett Gaussiskt filter på indatan för att bli av med de kraftiga diskontinuiteter vi har i vissa initialvärden. Eftersom vi kommer att ha ett konstant flöde i nästan alla problemställningar har vi dock låtit bli att göra så. Det hade varit trevligt att kunna filtrera datan inuti vår PDE-lösare, men det innebär en del fulhack i `MATLAB` och framför allt innebär det mer jobb än det är värt.

Gaussiska filter är enkla, både i praktiken och i teorin. De utgår från en Gaussfunktion (12) som sedan faltas med signalen man vill filtrera. Detta görs enkelt i `MATLAB` med hjälp av funktionen `conv`. Filtret är för övrigt ett s.k. *lågpassfilter*.

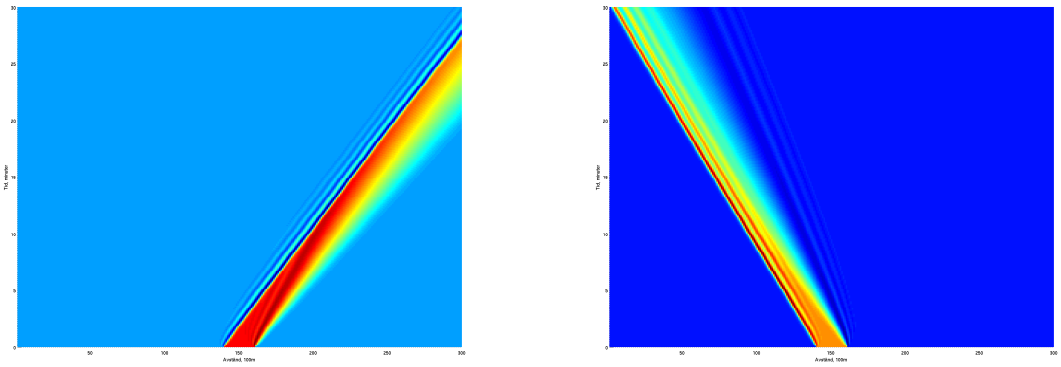
$$g(x) = \frac{e^{-\frac{x^2}{2\sigma^2}}}{\sqrt{2\pi\sigma^2}} \tag{12}$$

Vi har funnit att ett bra värde på standardavvikelsen σ är 0.15. Det är även värt att notera att vi klipper g och bara faltar den del av g som ligger i spannet $[-\frac{1}{2}, \frac{1}{2}]$.

Det är även värt att poängtera att detta filter faktiskt inte påverkar tolkningen av vår data, utan endast presentationen. Det kommer att filtrera bort en del extremvärden och artefakter, och "mjuka upp" den grafiska representationen av vår data, men det kommer inte att påverka det vi faktiskt tittar på (dvs. hur täthetssignaler rör sig, hur trafikljus introducerar täthetsvågor, och liknande), så att köra ett Gaussiskt filter på utdatan är helt försvarbart.

2.2.7 Vår metod

Efter en del analys (som vi går in på senare) valde vi att använda `pdepe`, tillsammans med det Gaussiska filter vi implementerat för att få lite finare utdata.



(a) $\rho < \frac{\rho_{max}}{2}$ (b) $\rho > \frac{\rho_{max}}{2}$

Figur 1: Lax-Wendroff-metoden vid två olika tätheter, över och under den kritiska gränsen.

3 Beräkningar

3.1 Metoder

För att lättare kunna bestämma oss för vilken metod vi skulle använda ställde vi upp ett enkelt problem och körde det igenom de metoder vi tittat på. Modellen vi använt för att testa metoderna är enkel; vi har en maxhastighet på 70 km/h och en maxdensitet på 200 bilar/km. I mitten har vi lagt in en störning i form av en markant ökning i densitet. Alla figurer visar densitet, med avstånd på x-axeln och tid på y-axeln. Förutom att den här modellen visar hur exakta metoderna är så visar den dessutom en intressant företeelse; våghastigheten beror på ρ på ett sådant sätt att densitetsvågen rör sig framåt när densiteten ligger under den kritiska densiteten, medans den rör sig bakåt när densiteten är högre än den kritiska densiteten. Det betyder i korthet att köer "rör sig" bakåt när det är mycket trafik, och framåt när det är lite trafik.

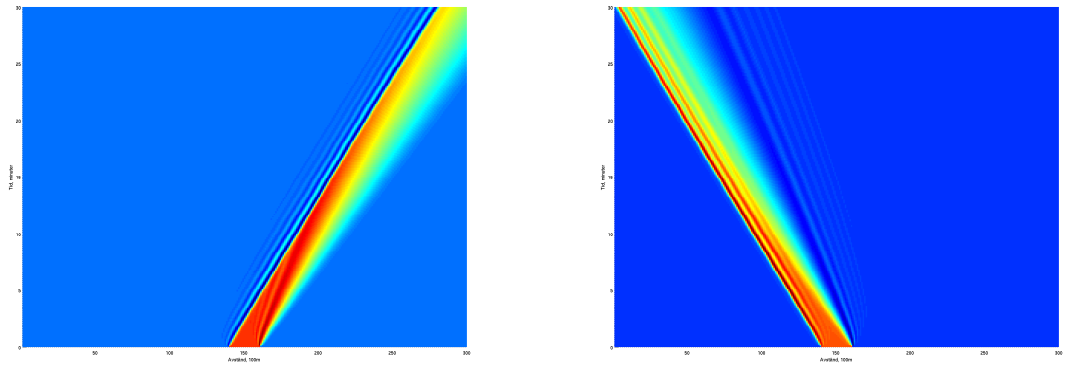
3.1.1 Lösning med `ode45` & `diff`

Den här metoden är relativt dålig. Den är mycket känslig för singulariteter och fungerar endast i undantag för vår modell. Vi gjorde ingen större analys av kombinationen `ode45` och `diff`, utan bestämde oss ganska tidigt för att gå in på lite bättre anpassade metoder. Att lösa PDE-problem med rena ODE-metoder är inte att rekommendera.

3.1.2 Lösning med Lax-Wendroff

Som man ser i Figur 1 är Lax-Wendroff en bra metod. Den är designad för just hyperboliska PDEer, och därmed väl lämpad för vår modell. Den har en nackdel, och det är att den inför ringartefakter i problemet, speciellt där derivatorna är stora. Detta manifesterar sig till exempel i Figur 1(b), och syns ganska tydligt.

Det är inget stort problem, men kan göra att modellen ser lite felaktig ut.



(a) $\rho < \frac{\rho_{max}}{2}$ (b) $\rho > \frac{\rho_{max}}{2}$

Figur 2: MacCormack-metoden vid två olika tätheter, över och under den kritiska gränsen.

3.1.3 Lösning med MacCormack

Som synes i Figur 2 är MacCormack en relativt exakt metod. Den är dessutom snabb, och enkel att implementera. Den har dock (precis som Lax-Wendroff) en nackdel, som främst visar sig i Figur 2(b): den introducerar ringartefakter av ungefär samma typ som de man stöter på i Fourierserier. Detta sker när derivatan av densiteten (med avseende på x) är stor. Detta dämpas till viss del när man alternerar mellan framåt- och bakåtdifferens, men kan inte tas bort helt.

Det är inget stort problem, men det gör att modellen i vissa situationer kan se lite udda ut.

3.1.4 Lösning med `pdepe`

Som tidigare nämnts kräver `pdepe` en hel del implementerande av funktioner för att fungera, men när den fungera ger den den helt klart mest exakta lösningen av de vi använt.

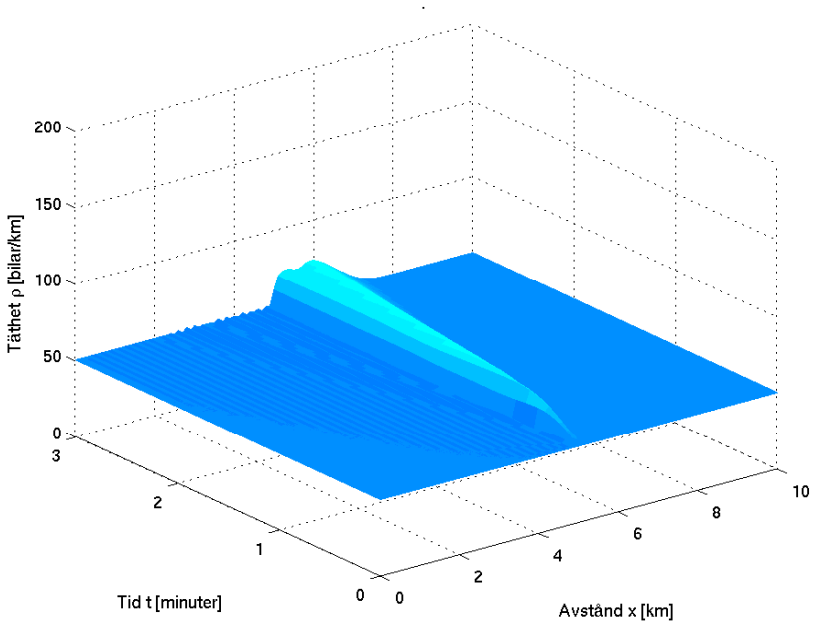
Metoden är även mycket snabbare (storleksordning 10 gånger snabbare än MacCormack), vilket gör att vi kan ha en mycket högre upplösning på vår lösning utan att beräkningstiden blir för lång.

De ringartefakter som dyker upp försvinner väldigt fint efter att man låtit datan passera genom ett Gausiskt filter och det går då lätt att urskilja trafikflödets beteende ur våra plottar.

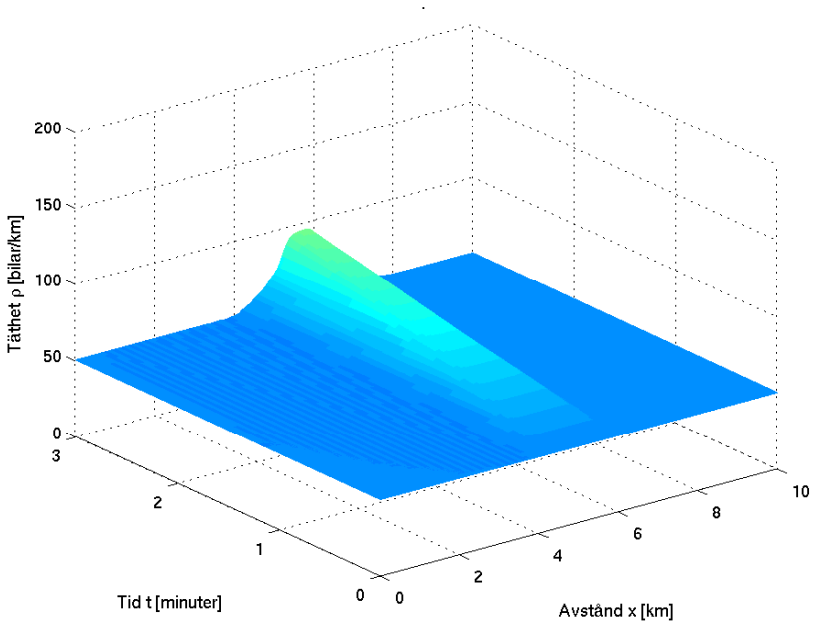
3.2 Lösningar till problemställningar

Efter att snabbt ha analyserat ett par olika möjligheter att ställa upp intressanta problem kom vi fram till ett par olika fall vi bestämde oss för att modellera. Dessa följer nu, tillsammans med figurer och korta kommentarer. Som sagt bestämde vi oss för att använda `pdepe` för att beräkna det PDE-system vi ställt upp.

Enkel rak väg med hastighetsförändring
En enkel hastighetsförändring kan modelleras genom att sänka v_{max}. I Figur 3 har vi sänkt hastigheten progressivt (linjärt) mellan $x = 4$km och $x = 6$km, från 110km/h ner till två olika värden. Som synes påverkas trafiken mer av en drastisk förändring, vilket är föga överraskande.

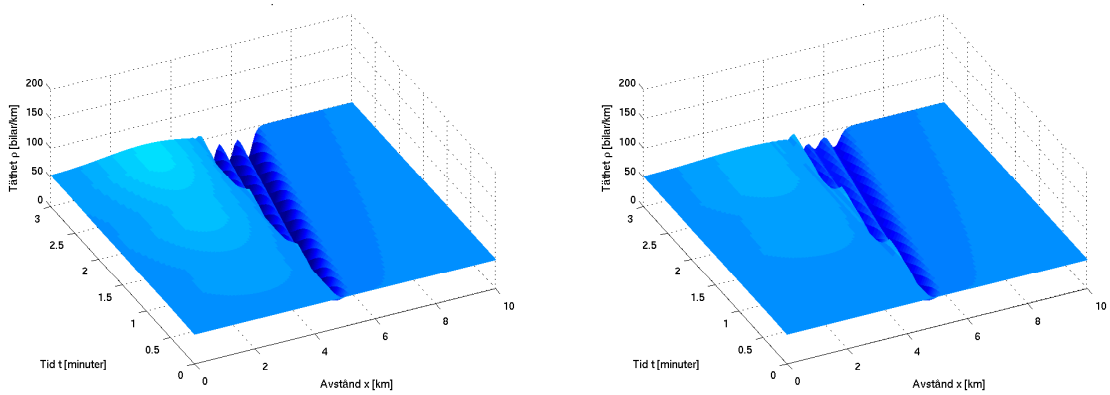


(a) Hastighetsminskning från 110km/h till 90km/h



(b) Hastighetsminskning från 110km/h till 70km/h

Figur 3: Modellering av en enkel hastighetsförändring runt $x = 5$km



(a) Övervägande rött ljus

(b) Övervägande grönt ljus

Figur 4: Ett enkelt periodiskt rödljus på en rak väg, $v_{max} = 50\text{km/h}$

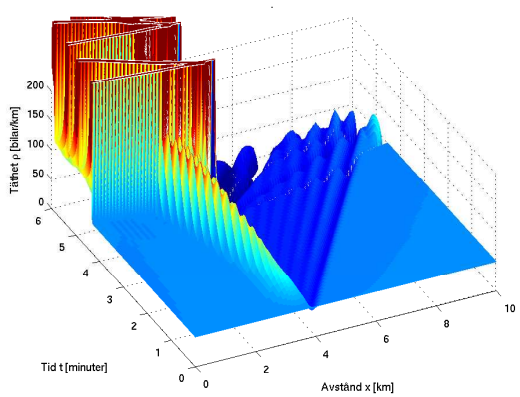
Enkel rak väg med trafikljus

I Figur 4 syns två olika periodiska rödljus med mer eller mindre än 50% rött ljus, tidsmässigt. Som man kan se byggs köer upp bakom det rödljus som har övervägande rött, vilket inte är särskilt förvånande. Beroende på mängden trafik från början måste man justera mängden rött ljus för att undvika köer; ju tätare trafik desto mindre rött. Man ser även djupa gropar efter trafikljuset. Detta beror på att trafikljuset stoppar trafik, och därmed kommer tätheten strax efter trafikljuset vara noll under den till trafikljuset visar rött. Inte heller detta är särskilt överraskande.

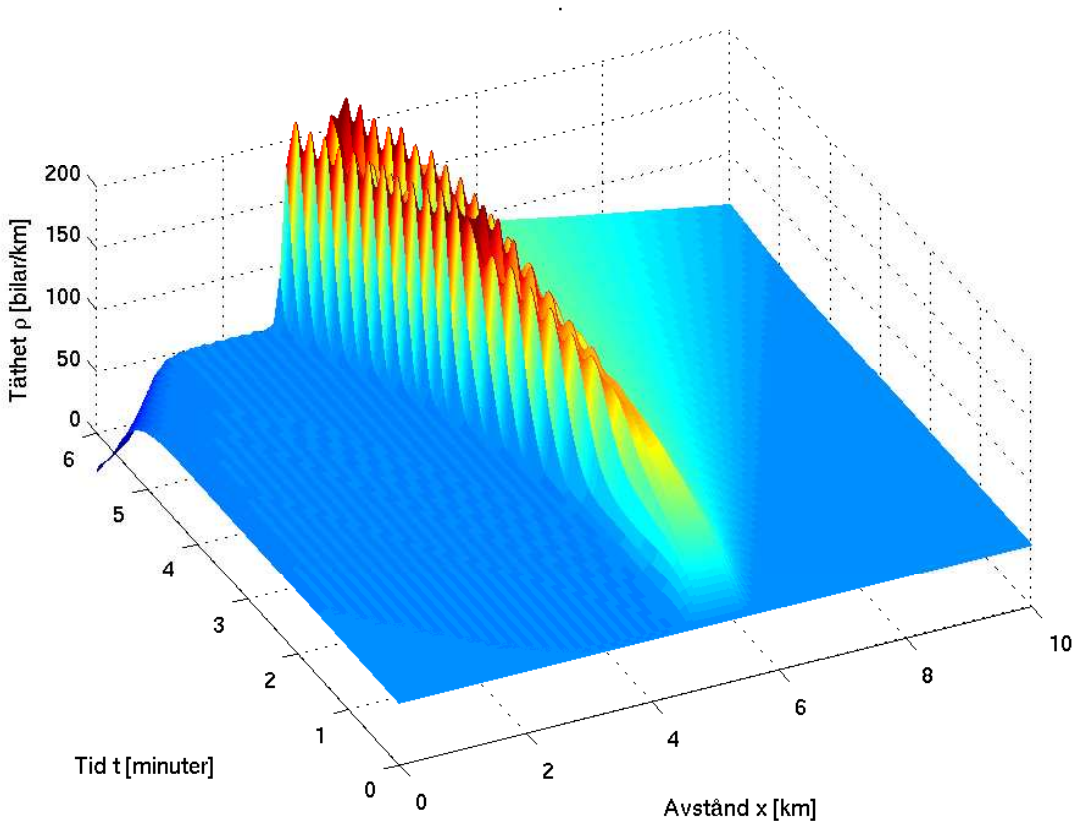
Enkel rak väg med filsammanslagning

En enkel filsammanslagning, dvs. en halvering av ρ_{max}, känns inte som något som bör ställa till problem. Vår modell visar sig dock vara ytters olämpad för sådana saker, vilket syns i Figur 5 — `pdepe` lyckas inte med att beräkna någon vettig lösning till PDE-systemet, utan fallerar efter en tid.

Det man förväntar sig se är en uppgbyggnad av köer bakom filsammanslagningen (förutsatt att trafiktätheten är så hög att den efter sammanslagningen överskrider $\rho_{max}/2$), vilket man kan se en anstymmelse till i vår modell. När den kommer upp i ρ_{max} går saker lite sämre, och hela modellen kollapsar.



Figur 5: Vår modell klarar uppenbarligen inte av allt. Här syns vad som skulle vara en filsammanslagning.



Figur 6: En enkel T-korsning utan trafikljus ger upphov till intressanta mönster

T-korsning; sammanslagning utan trafikljus

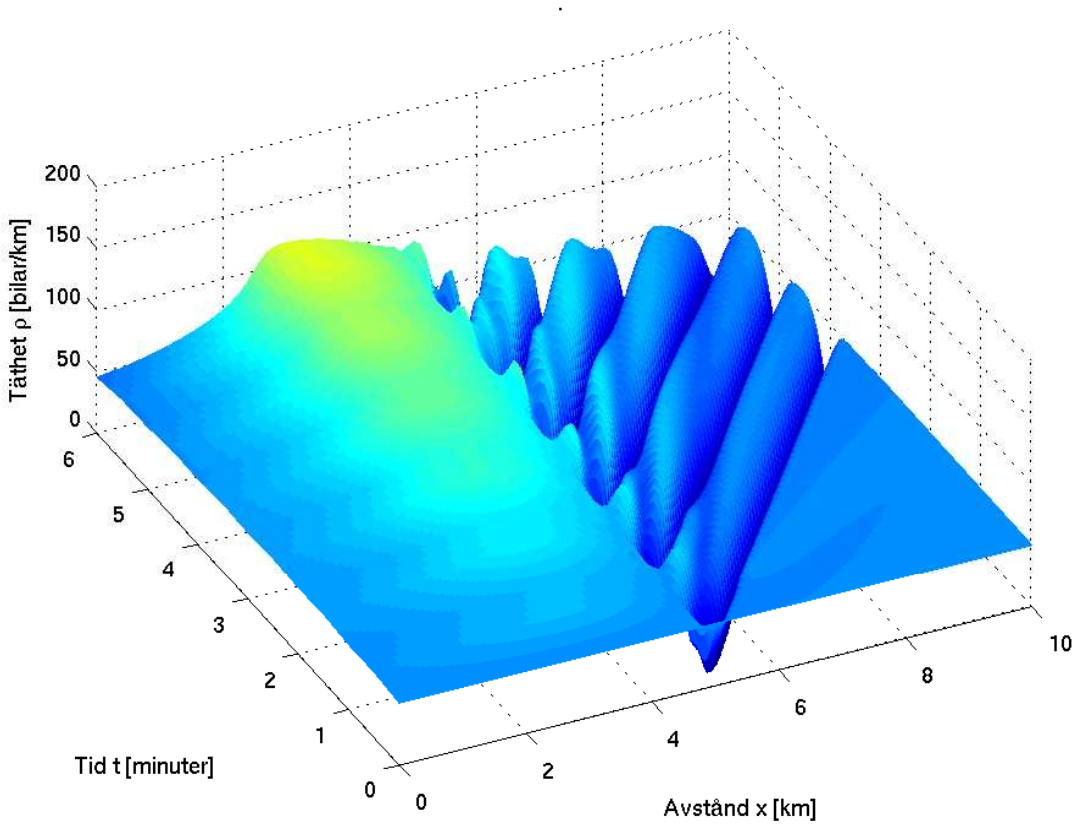
En enkel T-korsning utan trafikljus kan ses i Figur 6, där vi använder en flödeskälla för att injicera bilar till flödet. Detta ger samma effekt som att ha en sidoväg från vilken bilar släpps in. Som figuren visar ger detta upphov till lite intressanta köbildningar; köer bildas i impulser istället för att bildas som en enda lång kö. Framåt sker en täthetsökning som tunnas ut med tiden, precis som förväntat.

Det kan nämnas att mängden trafik vi lagt till i denna simulering är lika stor som den vi redan har. Vi kommer alltså i princip att fördubbla tätheten, vilket gör att vi kommer över den *kritiska gränsen*, och därmed bildas köer med negativ signalhastighet.

T-korsning; sammanslagning med trafikljus

Nu kommer vi till mer intressanta modeller. En enkel sammanslagning med trafikljus modellerar vi på samma sätt som en utan, men med ett periodiskt trafikljus inlagt, samt en periodicitet på flödeskällan som lägger till bilar.

I Figur 7 syns en T-korsning med trafikljus. Det ser i princip ut som en superposition av en vanlig T-korsning och ett trafikljus, vilket är precis vad det är. Föga överraskande, alltså. Det ger däremot ett väldigt intressant mönster i de toppar som bildas när trafikljuset visar grönt.

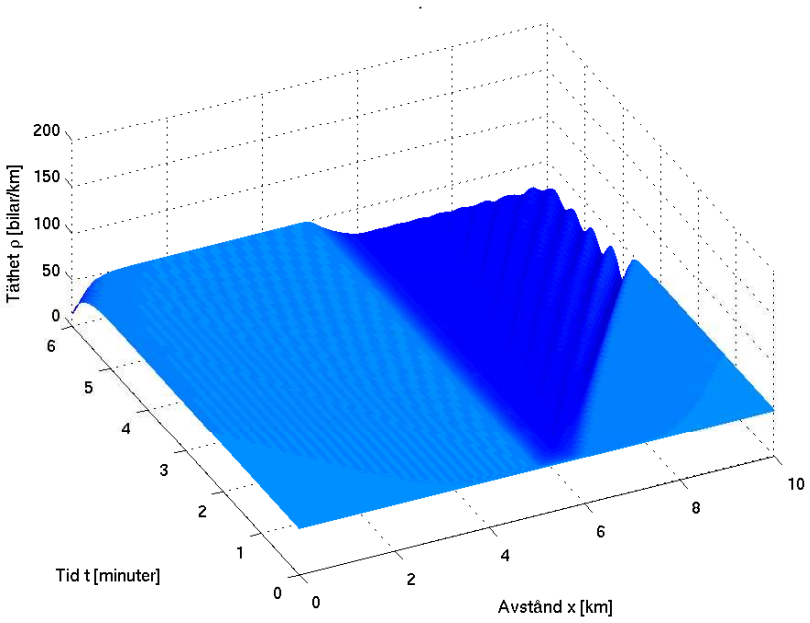


Figur 7: En T-korsning med trafikljus. Trafiken som släpps på har en lägre täthet än trafiken på vägen.

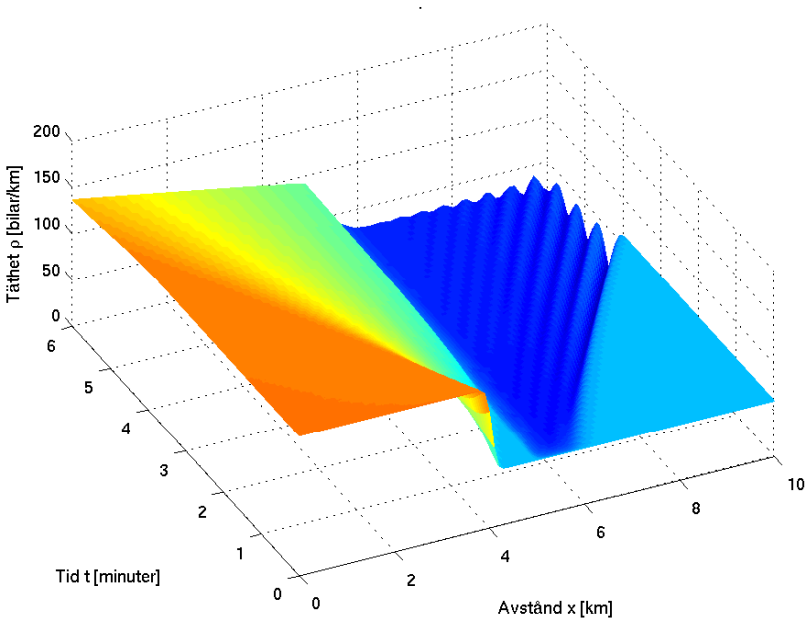
Enkel avfart
I Figur 8 ser vi en avfart där en majoritet av bilisterna väljer att svänga av. Vi låga trafiktätheter, som i Figur 8(a), märks ingen större skillnad. Vid större tätheter, som i Figur 8(b), syns skillnaden mycket mer. Om man jämför denna figur med en där (fyrkants)vågen tillåts avta utan några förändringar på Q så ser man att den naturliga sluttningen som bildas klipps av vid avfarten. Detta är inte oväntat, tätheten reduceras ju med 95%.

T-korsning; normal med trafikljus
I en T-korsning med normal trafik kommer antingen trafiken öka som i Figur 6 eller avta som i Figur 8, med variationer i graden bilar som tillkommer respektive försvinner ut ur systemet.

Fyrvägskorsning med trafikljus
En fyrvägskorsning är precis som tidigare nämnt egentligen bara en T-korsning med lite större flöden in eller ut, vilket gör att modelleringen av den inte tillför något nytt.



(a) Konstant flöde till avfarten



(b) Stor täthetsvåg möter avfarten

Figur 8: En avfart där 95% av bilisterna väljer att svänga av

Motorvägsavfart
Med den modellen vi använder skulle en motorvägspåfart (som inkluiderar en bit av motorvägen efter påfarten) bli en vägsträcka med hastighetsförändring (förslagsvis 90km/h till 110km/h), en T-korsning med flöde in i vårt system och sedan en vanlig vägsträcka med konstant hastighet men betydligt större tillåten täthet (flera filer). Vår modell skulle kräva att summan av tätheten på påfarten och motorvägen överstiger den maximalt tillåtna tätheten på motorvägen.

4 MATLAB-kod

Listing 1: Representation av funktionen Q i ode45, Lax-Wendroff och MacCormack

```
function out = Q(rho, t)
    rho_max = ones(size(rho))*20;
    v_max = ones(size(rho))*60/6;
    out = (rho .* v_max .* (1 - rho ./ rho_max));
end
```

Listing 2: Representation av funktionen Q i pdepe

```
function q=Q(rho,x,t)
  % [vmax] = km/minut
  vmax_0=50/60;
  vmax=vmax_0*ones(size(rho));
  % Modifikationer av Q (trafikljus etc)

  % Inga fler modifikationer
  q=vmax.*rho.*(1-rho./rhomax);
end
```

4.1 ode45 & diff

Listing 3: Funktionen som körs genom ode45

```
function rhot = modell3(t, rho)
    Qp      = diff(Q(rho, t));
    Qp      = [Qp(1);Qp];
    Qp(end) = Qp(end-1);
    rhot = -Qp;
end
```

4.2 Lax-Wendroff

Listing 4: Vår implementation av Lax-Wendroff-metoden

```
function [t, out] = LaxWendroff(in, tspan, rhs)
    if(size(in,2) ~= 1); in = in'; end
    deltat = 1/16;
    deltax = 1;
    u      = in;
    halfp  = [];
    unext  = [];
    out    = in;
    t      = tspan(1);
    for(i = tspan(1):deltat:tspan(2))
        % Lax step
        halfp = -deltat/(2*deltax)*(rhs([u(2:end);u(end)], i) - rhs(u,
           i)) + (1/2)*([u(2:end);u(end)] + u);
        % Wendroff step
        unext = -deltat/deltax*(rhs(halfp, i) -
           rhs([halfp(1);halfp(1:end-1)], i)) + u;
```

```
        % Output
        out = [out unext];
        u = unext;
        t = [t i+deltat];
    end
end
```

4.3 MacCormack

Listing 5: Vår implementation av MacCormack-metoden

```
function [t, out] = mccormack(in, tspan, rhs)
    if(size(in,2) ~= 1); in = in'; end
    deltat = 1/16;
    deltax = 1;
    u      = in;
    upred  = [];
    unext  = [];
    out    = in;
    t      = tspan(1);
    altern = 0;
    for(i = tspan(1):deltat:tspan(2))
        if(mod(altern,2) == 0)
            % Predictor
            upred = u - deltat/deltax*(rhs([u(2:end);u(end)], i) -
               rhs(u, i));
            % Corrector
            unext = (u + upred)/2 - deltat/(2*deltax)*(rhs(upred, i) -
               rhs([upred(1);upred(1:end-1)], i));
        else
            % Predictor
            upred = u - deltat/deltax*(rhs(u, i) -
               rhs([u(1);u(1:end-1)], i));
            % Corrector
            unext = (u + upred)/2 -
               deltat/(2*deltax)*(rhs([upred(2:end);upred(end)], i) -
               rhs(upred, i));
        end
        % Output
        out = [out unext];
        u = unext;
        t = [t i+deltat];
        altern = altern + 1;
    end
end
```

4.4 pdepe

Listing 6: Funktionerna som skickas till pdepe

```
% PDE-funktion (pdefun.m)
function [c,q,s] = pdefun(x,t,rho,dudx)
  c=-1;
  s=0;
  q=Q(rho,x,t);
end

% Begynnelsevillkorsfunktion (icfun.m)
function u = icfun(x)
  u=50;
end

% Randvillkorsfunktion (bcfun.m)
function [pl,ql,pr,qr] = bcfun(xl,rhol,xr,rhor,t)
  Ql=Q(rhol,xl,t);
  Qr=Q(rhor,xr,t);
  pl=Ql;
  ql=-1;
  pr=Qr;
  qr=-1;
end

% Anrop till pdepe (se MATLAB-help om pdepe):
pdepe(0, @pdefun, @icfun, @bcfun, xmesh, tspan);
```

5 Referenser

Diehl, S. (2006) *Modellering av trafikflöde*. http://www.maths.lth.se/matematiklth/personal/diehl/Modellering_trafikflode.pdf (2010-05-08).

Lax, P. och Wendroff, B. (1960) Systems of Conservation Laws. *Communications on Pura and Applied Mathematics*, vol. 13, nr 2, ss. 217–237.

MacCormack, R. W. (1969) *The effect of viscosity in hypervelocity impact cratering.* New York: AIAA.