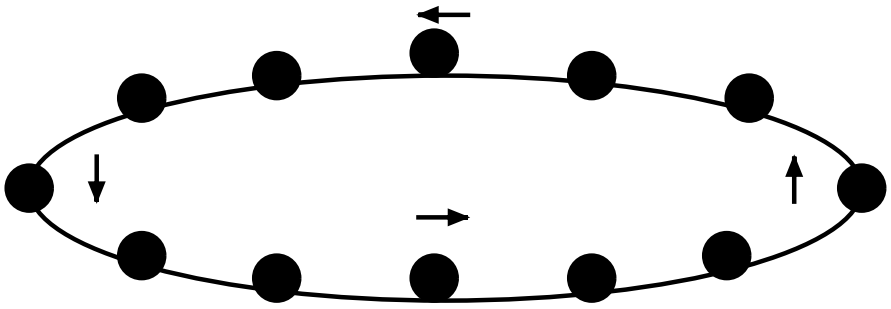
信号機のある
交通渋滞のシミュレーション



HE研　木村 泰大

2002年2月26日

目 次

謝辞

付録

1　はじめに

車を持っている人なら一度は交通渋滞に引っかかったことがあるはずである。狭い日本では今日もあちらこちらで渋滞が起こり、たくさんの人をいらいらさせている。そんな日本に住む私たちにとって、交通渋滞についていろいろ研究してみるということは、大変面白いことではないだろうか。

交通渋滞の原因というと、のろのろ運転、交通事故、信号、道路工事といったことを想像するのではないだろうか。これらが交通渋滞の大部分を占めているのは確かである。

しかし、それ以外にも小さな乱れによって生じる渋滞が存在すると考えられている。これは高速道路などでしばし観測されるのだが、高速道路を何台かの車が等間隔を維持しながら走ることは困難で、ちょっとしたブレーキが小さな乱れとなる。このような小さな乱れが、徐々に影響を与え、やがて渋滞を作ってしまうのである。

小さな乱れが原因の渋滞は、これまでにさまざまな人によって研究され、いろいろなモデルが考えられた。

2　モデル

代表的なモデル３つを取り上げそのモデルの説明、解析を行なう。

2.1　古典的な追従モデル

交通流のモデルの１つに追従モデルがある。[1] 追従モデルの特徴は、車の速度変化は前の車との相対速度によって決まることである。ようするに、前の車が速度を下げれば自分もブレーキを踏み速度を下げ、前の車が速度をあげれば自分もアクセルを踏み速度をあげ前の車に追従するというモデルである。

初期のころの追従モデルに関する論文として Pipes によって書かれたものがある。Pipes は、車の運動を現象的に考え、それが満たすべき方程式を導いた。一次元の追い越しがない状況で、車の理想的な車間距離は、ある一定時間 T で車が進める距離と止まっているとき運転手が自然にとる車間距離との和で表されると考える。k 番めの車の位置を x_k とし、位置の時間微分を表す $\dot{x}_k$ で走っているとすると、時間 T の間に進める距離は $T \cdot v_k$ になる。自然にとる車間距離 b を、前の車の位置を x_{k-1}、車の大きさを L_k とすると、前との車間距離は、

$$x_{k-1} - x_k = b + T \cdot \dot{x}_k - L_k \tag{1}$$

と表さる。(1) の時間微分をとると

$$\ddot{x}_k = \frac{\dot{x}_{k-1} - \dot{x}_k}{T} \tag{2}$$

となり、これは物理的な運動方程式

$$m\frac{d^2x_k}{dt^2} = \lambda(\frac{dx_{k-1}}{dt} - \frac{dx_k}{dt}) \tag{3}$$

と同値になる。これを車の数だけ連立させると、全体の運動方程式が得られる。簡単な解として $\dot{x}_k = 0$ すなわち、$\dot{x}_{k-1} - \dot{x}_k = 0 = \ddot{x}_k$ があるが、この場合において、小さな乱れとして $x_k = f_k e^{i\omega t}$ を式 (3) に代入し、解析をする。

$$i^2\omega^2 m f_k e^{i\omega t} = \lambda(i\omega f_{k-1}e^{i\omega t} - i\omega f_k e^{i\omega t})$$

式を変形して、

$$f_k = (1 + \frac{i\omega m}{\lambda})^{-1} f_{k-1} \tag{4}$$

一番先頭の振幅を $f_0 = 1$ とすると、k 番めの車の振幅の大きさは、

$$f_k = (1 + \frac{i\omega m}{\lambda})^{-k}$$

$$|f_k| = (1 + \frac{\omega^2 m^2}{\lambda^2})^{\frac{-k}{2}} \tag{5}$$

これから振幅が常に 1 以下で徐々に小さくなっていくので安定に近づいていくことがわかる。振幅と乱れは同じことであり、振幅が大きいほど乱れが大きいということである。つまり、このモデルでは先頭の車が乱れたとしても、後ろの車にいくほど乱れが小さくなので渋滞は起こらない。

しかしこのモデルは、前の車の速度変化を瞬時にとらえ、自分の速度を変化させるといったモデルであり、実際はそのような細かな運転はしない。つまり非現実的なモデルである。現実的には前の車が速度をだいぶ下げ、距離がつまったことを認知してからドライバーは速度を変化させるであろう。そこで Chandler は運転手の反応の遅れを導入し、次のような差分方程式にした。[2]

$$m\frac{d^2x_k(t)}{dt^2} = \lambda\{\frac{dx_{k-1}(t-\Delta)}{dt} - \frac{dx_k(t-\Delta)}{dt}\} \tag{6}$$

これを先ほどと同様に $x_k = f_k e^{i\omega t}$ を代入し、解析すると

$$|f_k| = \{1 + \frac{\omega^2 m^2}{\lambda^2} - \frac{2\omega m}{\lambda}\sin(\omega\Delta)\}^{\frac{-k}{2}} \tag{7}$$

中括弧の中が 1 より小さい場合は、後ろの車にいくほど振幅が大きくなる。よって $\frac{\omega m}{\lambda} > 2\sin(\omega\Delta)$ の時は乱れは小さくなっていき渋滞は起こらず、$\frac{\omega m}{\lambda} < 2\sin(\omega\Delta)$ の時は乱れは大きくなっていくので渋滞が起こる。

ここに挙げた 2 つのモデルは、速度変化は相対速度にしかよらない。そのため前の車との距離が充分大きく離れていたとしても、前の車が速度を変えたら自分も速度を変えるするという非現実的なことが起こる。そこで相対速度の代わりに、車間距離に影響を受けるというモデルが考えられた。

2.2 OV(Optimum Velocity) モデル

坂東らによって提案されたモデル（OV モデル）[3] では車の速度変化は、

$$\ddot{x}_n(t) = a\{U_{(x_{n+1}(t)-x_n(t))} - \dot{x}_n(t)\} \tag{8}$$

$$n = (1, 2, 3, \cdots, N)$$

という微分方程式で表される。N は車の総数、U は前の車との距離（$\Delta x = x_{n+1} - x_n$）で決まる関数、a はドライバーの感度を表す定数で、感度の値が大きいほど急加速急ブレーキをする。現実の世界ではドライバーは前の車との距離が短い場合には衝突を防ぐため減速し、距離が十分あれば、制限速度 U^{max} の範囲で速度を上げるということをする。したがって関数 U には次のような特性を持たせる必要がある。

1. 単調的に増加する関数
2. $U(\Delta x)$ は制限速度 U^{max} を持つ

車が等間隔、等速度で流れている状態から、ある 1 台の車が少し乱れると交通流はどのようになるか、を考える。まず、小さな乱れがない場合を考える。コースの長さを L、車の台数を N 台とすると車間距離は、

$$x_{n+1}(t) - x_n(t) = \frac{L}{N} \tag{9}$$

加速度は 0 なので、式 (8) の左辺をゼロとして解くと、

$$x_n(t) = U(\frac{L}{N})t + \frac{L}{N}n \tag{10}$$

が得らる。ここに小さな乱れとして、ゆらぎを $y_n(t)$ として導入し、ゆらぎが小さいとすると、

$$x_n(t) = U(\frac{L}{N})t + \frac{L}{N}n + y_n(t)$$

となり、$U'(\frac{L}{N}) = \frac{dU(\Delta x)}{\Delta x}|_{\Delta x=\frac{L}{N}}$ と定義すると、

$$\ddot{y}_n(t) = aU'(\frac{L}{N})(y_{n+1}(t) - y_n(t)) - \dot{y}_n(t) \tag{11}$$

となる。これに $y_n(t) = f_n e^{i\omega t}$ を代入して安定性を調べると、

$$f_n = \{\frac{U'(\frac{L}{N})}{U'(\frac{L}{N}) - \omega^2 + i\omega a}\} f_{n+1} \tag{12}$$

となりこの係数の絶対値が 1 より小さければ、前の車よりゆらぎが小さくなるので安定していく。

$$|\frac{U'(\frac{L}{N})}{U'(\frac{L}{N}) - \omega^2 + i\omega a}| < 1$$

$$\frac{\omega^2}{a^2} - \frac{2U'(\frac{L}{N})}{a} + 1 > 0 \tag{13}$$

で、$\omega \to 0$ の場合次のようになる。

1. $\frac{2U'(\frac{L}{N})}{a} < 1 \cdots$ 安定である。（渋滞は起こらない）
2. $\frac{2U'(\frac{L}{N})}{a} = 1 \cdots$ 臨界の状態である。
3. $\frac{2U'(\frac{L}{N})}{a} > 1 \cdots$ 不安定である。（渋滞が起こる）

2.3 UV モデル

早川 . 中西により提案されたモデル（UV モデル）[4] は、OV モデルに後方の車からの影響を加えたモデルである。後ろの車が近付いてくると心理的に速度を上げてしまうということを考慮に入れている。式で表すと、

$$\ddot{x}_n(t) = a\{U_{(x_{n+1}(t)-x_n(t))} \cdot V_{(x_n(t)-x_{n-1}(t))} - \dot{x}_n(t)\} \tag{14}$$

$$n = (1, 2, 3, \cdots, N)$$

となり、OV モデルの関数 $U(\Delta x)$ に後ろの車との距離（$\Delta x2 = x_n(t) - x_{n-1}(t)$）による関数 $V(\Delta x2)$ を掛け合わせたものになっている。関数 $V(\Delta x2)$ は、後ろの車との距離が詰まると OV モデルより加速効果を上げるため 1 より大きくする必要があり、距離が十分ある時は、OV モデルと変わらないようにするため 1 に漸近するようにしなければならない。また単調減少関数にする必要がある。

このモデルの解析も行なわれている。[5]

3　UV モデルのプログラムの改良

UV モデルに基づいて作られたプログラムに問題点があり、1 車線なので本来は追い越すことのできないはずの車が前の車を追い越すという現象が起こる。シミュレーション上では追い越すだけで済むが、これが現実の世界ならば、車一台分しかない道で車が後ろから突っ込んでくることになるのだから事故になる。まずはこの原因を考え、プログラムを改良する。

3.1　シミュレーション環境

シミュレーション環境は次のとうりである。　　1 周２００の長さの円形のサーキットに１００台の車を等間隔に並べ一斉に同じ方向にスタートさせる。1 車線で前の車を追い越すことはできない。ドライバーはみんな同じ感度を持っている。感度が同じということは、車ごとに、運転に差がないということで、条件が同じならばどの車も同じ反応を示す。ある時刻にスタートした車は、

1. 自分の速度
2. 前の車との車間距離
3. 後ろの車との車間距離（UV モデルのみ）

から計算して、次の時刻の速度と位置を決める。

3.2　後ろの車の影響を変える

OV モデルを基にして作られたプログラムでは事故がなく動いたので、OV モデルと UV モデルの違いである後ろの車との関係に問題がある可能性がある。

UV モデルにおいて車の速度変化は前にも述べたように、

$$\ddot{x} = a\{U(\Delta x) \cdot V(\Delta x2) - \dot{x}\} \tag{15}$$

という微分方程式で表される。この値が正ならば加速をし負ならば減速をする。例えば現在の速度が 3 だとすると、$U \cdot V$ が 4 であれば加速をし、2 であれば減速をする。$U \cdot V$ の値が大きい方が、加速をしやすいことになる。U は前の車との距離 dx で変わる関数で、

$$U = \tanh(\Delta x - 2) + \tanh(2) \tag{16}$$

V は、後ろの車との距離で変わる関数で、

$$V = 1.0 + \frac{1.0 - \tanh(\Delta x2 - 2)}{1.0 + \tanh(2)} \tag{17}$$

を事故のあるプログラムでは使っていた。図 1 に U を Δx の関数として図 2 に V を $\Delta x2$ の関数として示す。また図 4 には、$U \cdot V$ を Δx の関数として示す。図 4 で OV モデルは、関数 V の影響がないので図 1 と変わらない。UV モデルで、$\Delta x2 = 0.1$ とほとんど後ろの車との距離がない場合は、OV モデルと比べて 2 倍になっている。後ろから煽られて急がされたとしても、これほどまで速度変化に影響を与えないはずだ。１周２００のコースで１００台の車を等間隔に走らすと車間距離は 2 になる。であるから、後ろの車との距離が 2 というのは距離が開いているといえる。このような場合でも、OV モデルに対して 1.5 倍になっている。これでは煽りに対する影響が強過ぎ前の車に衝突してしまってもしかたないのではないだろうか。

そこで、関数 V を次のように変え、影響を弱めてみる。

$$V = 1.0 + \frac{1.0 - \tanh(\Delta x2 - 1.3)}{2\{1.0 + \tanh(1.3)\}} \tag{18}$$

図 3 に変更後の V を $\Delta x2$ の関数として示す。図 1 と図 3 を比較するとわかるが、変更前に比べて後ろの車との距離がない時の影響を弱め、関数 V の影響がなくなる後ろの車との距離を短くしている。図 5 に V を変えた後の $U \cdot V$ を Δx の関数として示す。このように変えてシミュレーションを行なったところ、前の車を追い越す現象はなくなり問題は解決した。

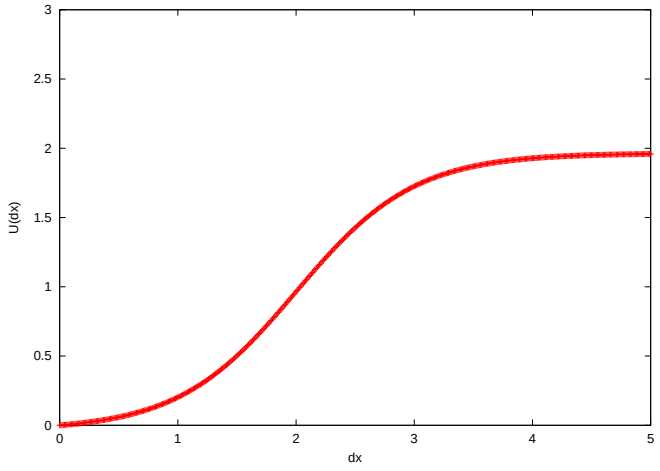


図 1: U vs Δx

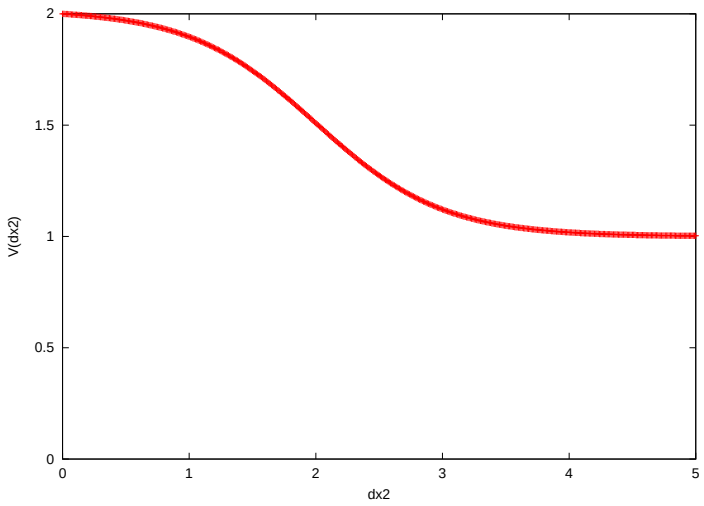


図 2: V vs Δx2

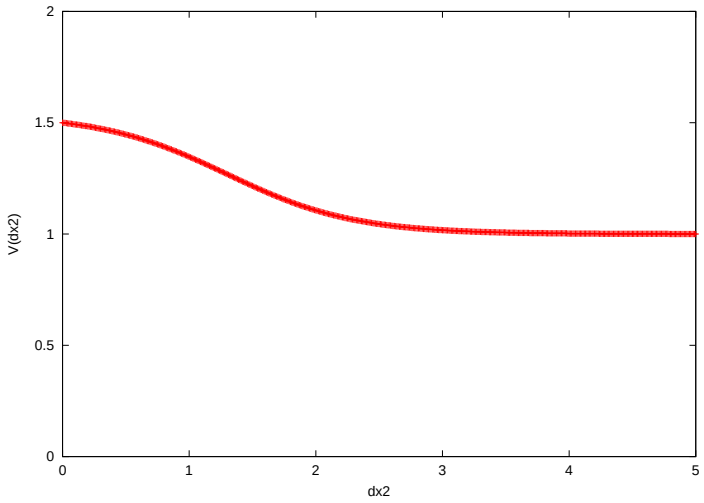


図 3: 改良後の V vs Δx2

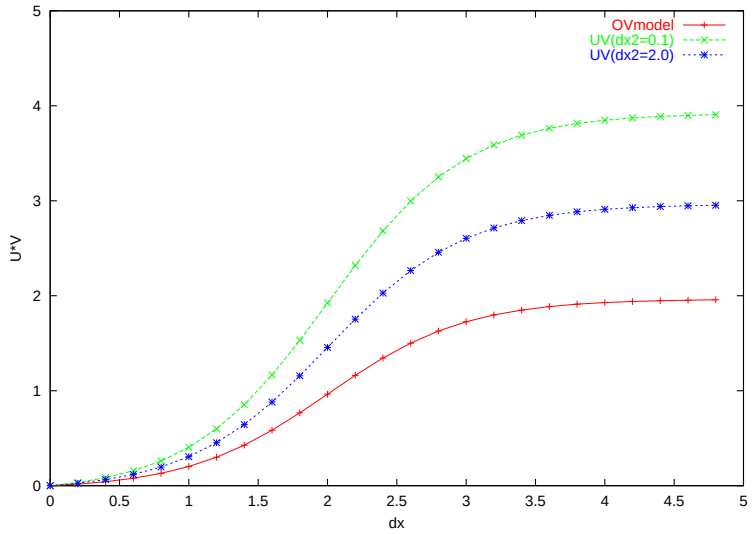


図 4: U·V の Δx,Δx2 依存性

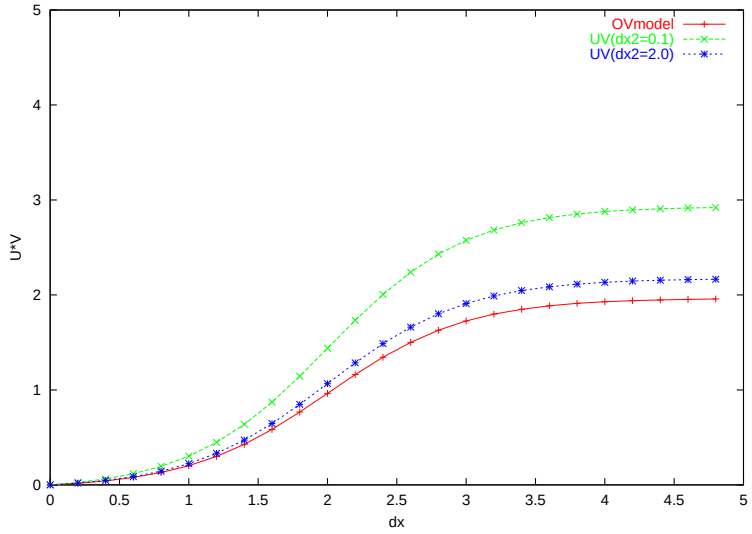


図 5: 改良後の U·V の Δx,Δx2 依存性

4　信号機のあるプログラム

複雑な交通網において信号機はなくてはならないものである。信号があるおかげで私たちは安全な交通社会にいることができる。しかし信号によって渋滞が起きる場合もあり信号をつけることで、デメリットも生じる。そこで、信号が交通流にどのように影響を与えるかをシミュレーションし、検証する。

4.1　渋滞非渋滞のグラフ

シミュレーション上で渋滞が発生したかどうかは時間に対する車の位置のグラフから判断できる。渋滞の中では、時間だけが経ち車の位置がほとんど変わらない。横軸に時間、縦軸に車の位置をとった場合、線の傾きが緩い場所は時間だけが経ち車の位置が変わっていないので渋滞が起きていると判断できる。

図 6 と図 7 に、ある 1 台の、時間における車の位置の変化を示す。1 周が２００の円形サーキットなので位置２００と位置 0 は同じ位置で、線が複数あるのは 1 台の車が円形サーキットを複数周まわっているからである。図 6 は、常に等速度で走っていて渋滞がない状態を示している。それに対して図 7 は線の傾きが緩く、位置がほとんど変わらず時間だけ経っている場所があり、渋滞が発生している状態を示している。

4.2　信号機の導入

次に、シミュレーションに信号機を導入する。簡単な場合を考え、これまでに使ってきた円形サーキットに 1ヶ所信号機を設置する。現実の世界から考えると、信号が青の時は右左折がない円形サーキット上では車に影響を与えないだろう。信号が赤の場合、自分が先頭ならば停止線との距離をみて速度を変化させ停止線で止まるだろう。先頭でない場合は前の車を見て速度を変化させ前の車が止まるとブレーキを踏み自分も止まるだろう。よって、信号が赤で自分が先頭車両の場合だけ、信号機との距離で速度を変化させるようにする。

また、現実の世界では黄信号があり、突然赤信号になったら急ブレーキを踏んでも止まれず教習所でもそのような場合は止まるなと教えられる。そこで、信号が青から赤に変わった時点で信号との距離があまりない場合はそのまま進ませ事故を防ぐようにする。

4.3　信号の影響による交通流の変化

改良したUVモデルでは、1周２００、車の台数１００台、感度1.5という状況では、車間距離２、速度１、を保ちながら等間隔に走る。渋滞がないこの条件で、０ステップ（ステップは時間に相当するもの）から５００ステップまで、信号を青の状態にして車を走らせ、５００ステップから１０００ステップまでの間信号を赤にし、１０００ステップからまた信号を青に戻す。そして、０ステップから１４０００ステップ付近までを観測する。図6や図7と同様にある車1台のステップに対する車の位置の変化を8から11に示す。

図8はスタート直後信号の影響を受けていない状態で等速度で進んでいるのがわる。もし信号がなければ、この条件では図8の状態が永遠に続く。図9は１０００ステップ付近で、信号が赤から青に変わった瞬間からしばらくを観測したものである。最初、前の車が走り出すのを待っていて進んでいない。少しして走り出すが車が1周する辺りで、時間だけが進むところがあり、渋滞が発生していることがわかる。図10はしばらくステップが経過した７０００ステップ付近の様子で、ほとんど元の状態に戻っているが、少し揺らぎがある。図11は、１３０００ステップ付近を示していて、ここまで経過すると元の状態に戻っている。

次に、信号の影響がなくなるまでの時間を測定するため、先ほどと同じ条件で、車間距離<１.９あるいは、 速度<０.９の場合、最初の状態と変わっているので、車は信号の影響を受けているとし、影響を受けている車の台数の変化を調べる。図12にステップごとの影響を受けている車の台数を示す。図12から元の状態に戻るまで（影響を受けている車の数が０になるまで）に赤信号の長さの約２５倍もかかっている。ここから信号は長い間交通流に影響を与えるということがいえる。

4.4　信号の長さと状態が元に戻るまでの長さの関係

次に、信号の長さと元の状態に戻るまでの時間にどのような関係があるかを調べる。先ほどの条件で、信号の長さだけを変え、信号の影響を受けている車の数が０になるまでのステップを計る。図13に、信号の長さに対する元の状態に戻るまでの長さを示す。信号が短いところで元に戻るまでのステップが急に増え、それからは信号が長くなっても元に戻るまでのステップはほとんど変わっていない。信号の長さよりも、信号があるかないかの方が交通流に影響を与えているということがいえる。

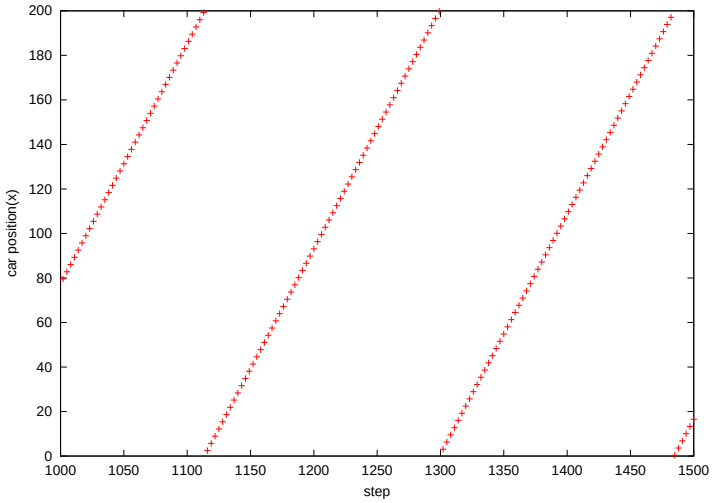


図 6: 渋滞のない場合の、時間毎の車の位置の変化

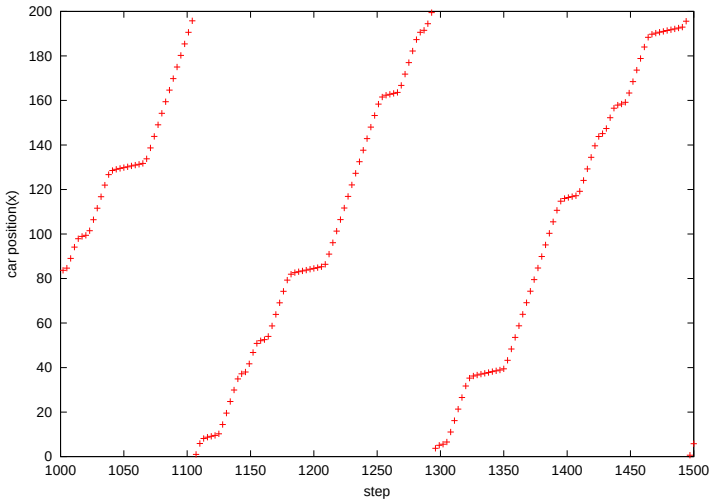


図 7: 渋滞のある場合の、時間毎の車の位置の変化

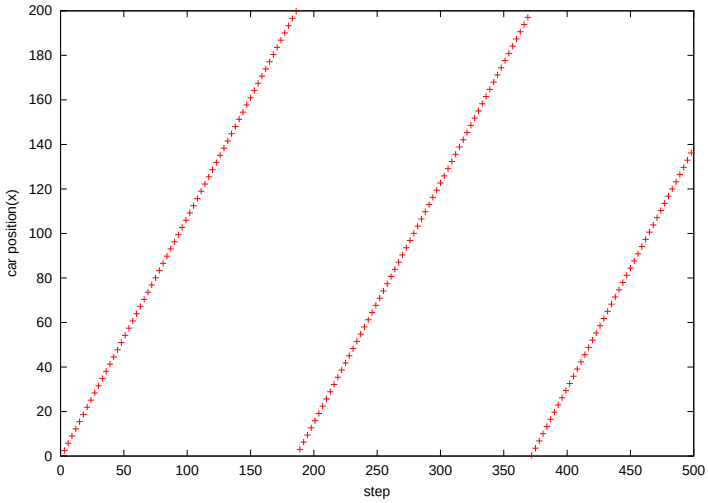


図 8: スタート付近の、時間毎の車の位置の変化

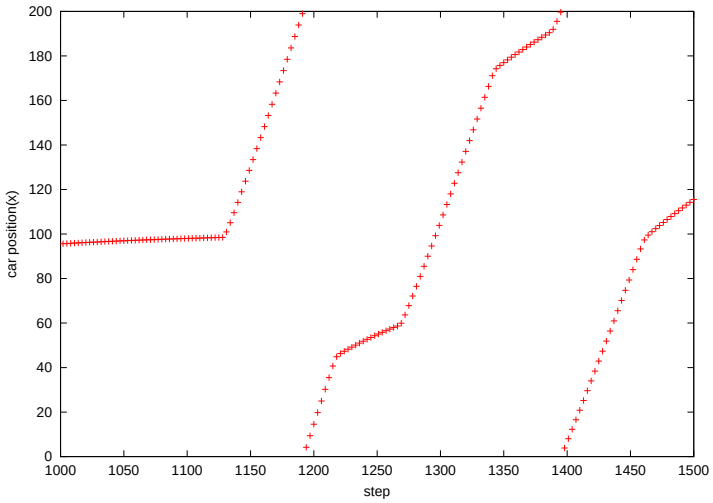


図 9: １０００ステップ付近の、時間毎の車の位置の変化

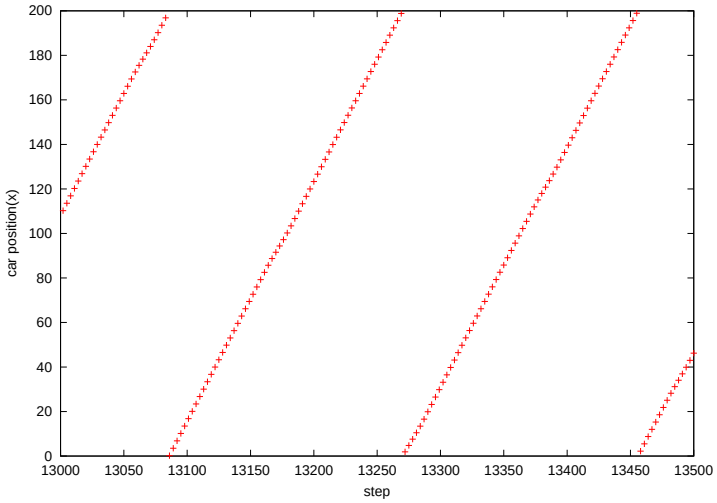


図 10: ７０００ステップ付近の、時間毎の車の位置の変化

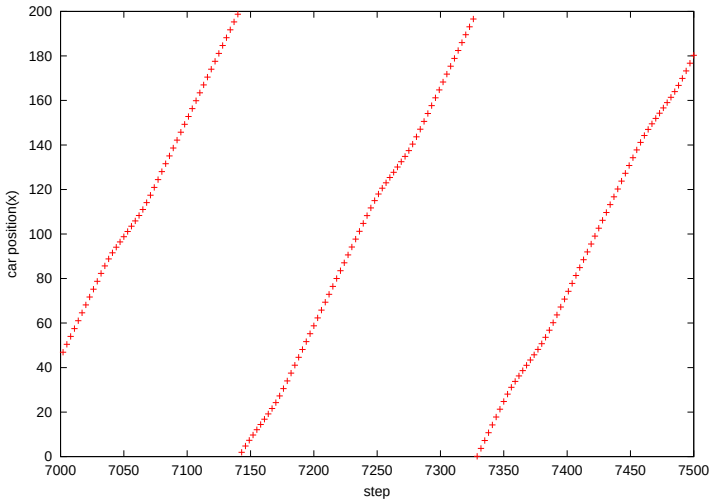


図 11: １３０００ステップ付近の、時間毎の車の位置の変化

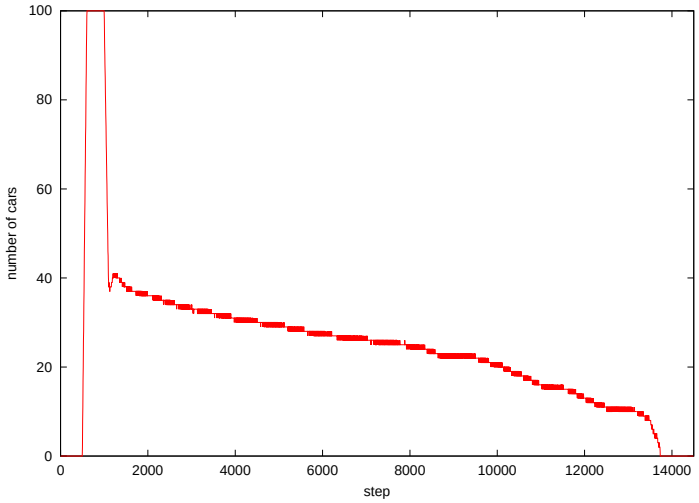


図 12: 影響を受けた車の数

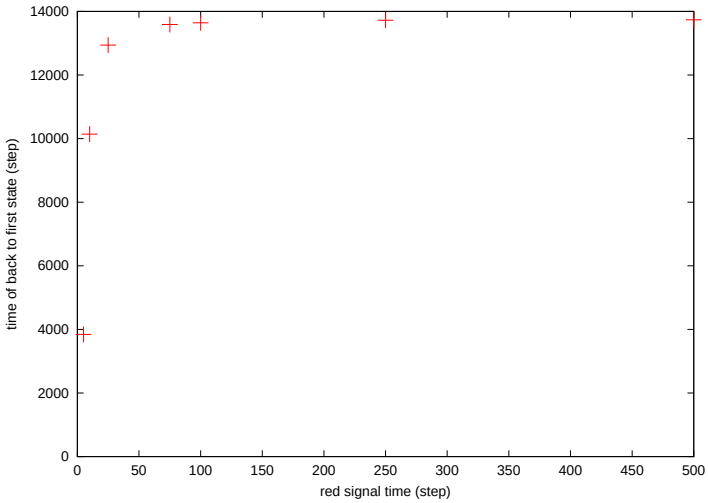


図 13: 信号の長さと元に戻るまでの時間の関係

5　まとめ

卒業研究において、代表的な３つのモデルの解析を行ない、渋滞が起こる条件と、渋滞が起こらない条件を調べた。UVモデルを基にしたプログラムの衝突の起こる原因を探り、後ろの車との距離による関数Vを改良し衝突が起こらないようにした。改良したプログラムに信号機を設置し交通流がどのように変化するかシミュレートし、次のことがわかった。。

渋滞のない1周２００、車の台数１００台、感度1.5という状況では、円形サーキットに１ヶ所だけ信号機設置し、青信号から１度だけ赤信号に変え、また青信号に戻すと次のようなことがいえる。

1. 信号があっても、いずれ元の状態に戻る。

2. 信号があると、その後長い間交通流に影響を与える。

3. 信号の長さよりも、信号があるかないかが交通流に影響を与える。

現実の世界では、信号機のある円形サーキットを走ることはまずなく、現実の世界では車は右左折を行なうので、この結果が現実の世界に直接結び付くとはいえない。実際、信号による渋滞は右左折による車の速度変化や交通量変化が主な原因である。また、円形サーキットと似た環境にある、右左折のない直線道路においても、ドライバーに個性があり、同じ感度ということはありえず、のろのろ運転など信号に関係のない渋滞も起こるのでこのような渋滞は、あまり実感することができないかもしれない。しかし今後、自動走行システムが実現していくにつれてドライバーの個性に差がなくなり、このような渋滞によっての交通流への影響が出てくるであろう。

それぞれが違った感度、最高速度を持ち、遅い車を抜くことのできる2車線プログラムを作ることにより、より現在の交通状況にあったシミュレーションを行なえるので、今後２車線プログラムを実現させたい。

最後に、長野県には、歩行者がほとんど通らないのにスクランブル交差点であったり、夜、車が1台も通らないのに押しボタン式でない場所がたくさんある。先ほども述べたように今後自動走行システムが実現するにつれて、このような信号による交通流への影響が顕著に出てくるであろう。信号機を設置する上で１番に考えるのは安全性であろう。しかし、交通流への影響も考える必要がある。信号機を設置する時は熟慮に熟慮を重ねて設置し、信号機を安易な考えで設置して交通流を妨げるということは渋滞を助長しドライバーのフラストレーションを満たし、しいては、事故につながりかねない危険を持つ。

参考文献

[1] L.A.Pipes: AN Operational Analysis of Traffic Dynamics

[2] R.E.Chandler,R.Herman,and E.W.Montroll: Traffic Dynamics:Studies in Car Following

[3] M.Bndo,K.Hasebe,A.Nakayama,A.Shibata and Y.Sugiyama: Stracture stability of congestion in traffic dinamics

[4] H.Hayakawa and K.Nakanishi: Theory of traffic jam in a one-line model

[5] 佐野 淳次: 追従 Model における一次元交通流の研究

謝辞

卒業研究においていろいろと御指導していただいた指導教官の竹下先生、長谷川先生、宮崎さんに深く御礼申し上げます。また、いろいろアドバイスをしてくれたHE研のみなさんに感謝します。コンピュータなどを共同で使い、いろいろサポートしてくれたHE研電子研合同研究室のみなさんありがとうございました。

付録

UVモデルを基にしたプログラム

```
ccc      program congestion
       program cong
       implicit none
       real time_init,time_last,time,step
       integer nnt,numb,nac,nstep,icar,roop,r,jam,istep,i
       integer ja(10000,1),s,fstep
       real x(150,2),xdata(150,10000),xperiod,dxdata(150,10000)
       real x1,x2,x1_next,x2_next,dx,dx2,ll,yfix,vdata(150,10000)
       real adata(150,10000)
       parameter (time_init=0.0,time_last=2000,step=0.06,ll=200.0)
       parameter (nnt=100)
       nac=0
       nstep=0
       istep=0
```

それぞれの変数を定義している。

```
c ---------------------------------  initialization
       do numb=0,nnt
        x(numb,1)=(ll/nnt)*(numb-1)
        x(numb,2)=0.0
         do i=1,10000
            xdata(numb,i)=0.0
            dxdata(numb,i)=0.0
            vdata(numb,i)=0.0
            adata(numb,i)=0.0
            ja(numb,1)=0
       enddo
       end do
```

データの初期化

```
        x(1,1)=x(1,1)+0.1
        time=time_init
```

小さな乱れとして先頭車両をずらしている。

```
c----------------------------------     main loop
        do while(time .lt. time_last)
         jam=0
```

```
          time=time+step
          nstep=nstep+1
c--------

          do numb=1,nnt
           x1=x(numb,1)
           x2=x(numb,2)

           if(numb.eq.nnt)then
           x(numb+1,1)=x(1,1)+ll
          endif

           dx=x(numb+1,1)-x(numb,1)

          if(numb.eq.1)then
           x(numb-1,1)=x(nnt,1)-ll
          endif

           dx2=x(numb,1)-x(numb-1,1)
           call calc_next_step(time,x1,x2,step,x1_next,x2_next,dx,dx2)
           x1=x1_next
           x2=x2_next
           x(numb,1)=x1
           x(numb,2)=x2
          end do
```

do の中で、前の車との距離、後ろの車との距離を計算し、後のルンゲクッタの 4 次法を使って次のステップでの位置、速度を決めている。 2 つの if の中は、境界条件を表している。

```
            if(nstep.eq.(nstep/50)*50) then
            istep=istep+1
```

情報が多いので、50nstep につき 1 回出力させる。

```
c------
               do icar=1,nnt
              xperiod=x(icar,1)
                do while(xperiod.gt.0)
                 xperiod=xperiod-ll
                enddo
```

円形サーキットでシミュレーションを行なっているといったが、実際は、いったん直線を走らせ、その後 1 周の長さで区切っていくことにより、円形サーキットを表現している。ここでは、 1 周の長さで区切ることを行なっている。

```
                        xdata(icar,istep)=xperiod+ll
                        vdata(icar,istep)=x(icar,2)
```

上のように区切ると区切り過ぎてしまうので、 1 周の長さをたすことにより補正している。出力用にデータを移す。

```
             end do
           endif
        end do

c-------------------------------------      end of simulation loop

           do istep=1,600
            do icar=1,1
            fstep=istep*3
```

出力する時間と車を決める。fstep というのは、実際のステップと同じにするための
補正。

```
             dxdata(icar,istep)=xdata(icar+1,istep)-xdata(icar,istep)
             dxdata(nnt,istep)=xdata(1,istep)-xdata(nnt,istep)
          if(dxdata(icar,istep).le.0)then
           dxdata(icar,istep)=dxdata(icar,istep)+ll
          endif
```

前にも車間距離を計算したが、データとしての車間距離を車の位置から計算している。
円形サーキットのため前の車が位置 1 にいて、後ろの車が位置 1 9 9 にいるといった場合もあるため補正している。

```
          adata(icar,istep)=(vdata(icar,istep)-vdata(icar,istep-1))/step
```

加速度を速度から計算している。

```
c-------------------------------------     print
           write(*,100) fstep,icar,xdata(icar,istep),dxdata
     a(icar,istep),vdata(icar,istep),adata(icar,istep)
```

```
100   format(i10,3x,i10,3x,f10.3,3x,f10.3,3x,f10.3,3x,f10.3)
```
ステップ、車のナンバー、位置、車間距離、速度、加速度を出力。
```
            enddo
          enddo



       stop
       end
c=============================================================
       subroutine calc_next_step(t,x1,x2,step,x1_next,x2_next,dx,dx2)
       implicit none
       real t,x1,x2,step
       real x1_next,x2_next,dx,dx2
       real diff_eq_1,diff_eq_2
       real k1_1,k2_1,k3_1,k4_1,k1_2,k2_2,k3_2,k4_2
        k1_1=step*diff_eq_1(t,x1,x2)
        k1_2=step*diff_eq_2(t,x1,x2,dx,dx2)
        k2_1=step*diff_eq_1(t+0.5*step,x1+0.5*k1_1,x2+0.5*k1_2)
        k2_2=step*diff_eq_2(t+0.5*step,x1+0.5*k1_1,x2+0.5*k1_2,dx,dx2)
        k3_1=step*diff_eq_1(t+0.5*step,x1+0.5*k2_1,x2+0.5*k2_2)
        k3_2=step*diff_eq_2(t+0.5*step,x1+0.5*k2_1,x2+0.5*k2_2,dx,dx2)
        k4_1=step*diff_eq_1(t+step,x1+k3_1,x2+k3_2)
        k4_2=step*diff_eq_2(t+step,x1+k3_1,x2+k3_2,dx,dx2)
        x1_next=x1+(k1_1+2.0*k2_1+2.0*k3_1+k4_1)/6.0
        x2_next=x2+(k1_2+2.0*k2_2+2.0*k3_2+k4_2)/6.0
       return
       end
```
ルンゲクッタの 4 次法
```

       real function diff_eq_1(t,x1,x2)
       implicit none
       real t,x1,x2
        diff_eq_1=x2
       return
       end

       real function diff_eq_2(t,x1,x2,dx,dx2)
```

```
      implicit none
      real t,x1,x2
      real legal,legal2,dx,dx2
       diff_eq_2=1.0*(legal(dx)*legal2(dx2)-x2)
      return
      end

      real function legal(dx)
      implicit none
      real dx,n
      parameter (n=2.0)
      legal=tanh(dx-n)+tanh(n)
      return
      end
```
前の車との距離による関数の設定

```
      real function legal2(dx2)
      implicit none
      real dx2, n
      parameter (n=1.3)
      legal2=1.0+(1.0-tanh(dx2-n))/(1.0+tanh(n))/2
```
後ろの車との距離による関数の設定

```
      return
      end
```

信号機のあるプログラム

```
ccc      program congestion
       program cong
       implicit none
       real time_init,time_last,time,step,i
       integer nnt,numb,nac,nstep,icar,roop,r,jam,istep
       integer ja(10000,1),s,fstep
       real x(150,2),xdata(150,10000),xperiod,dxdata(150,10000)
       real x1,x2,x1_next,x2_next,dx,dx2,ll,yfix,vdata(150,10000)
       real adata(150,10000)
       parameter (time_init=0.0,time_last=20000,step=0.06,ll=200.0)
       parameter (nnt=100)
       nac=0
       nstep=0
       istep=0
c ----------------------------------  initialization
       do numb=0,nnt
        x(numb,1)=(ll/nnt)*(numb-1)
        x(numb,2)=0.0
         do i=1,10000
            xdata(numb,i)=0.0
            dxdata(numb,i)=0.0
            vdata(numb,i)=0.0
            adata(numb,i)=0.0
            ja(numb,1)=0
       enddo
       end do
        x(1,1)=x(1,1)+0.1
        time=time_init
c----------------------------------     main loop
        do while(time .lt. time_last)
         jam=0
         time=time+step
         nstep=nstep+1
c------                car loop up to nnt
        if((time.ge.750).and.(time.lt.1000))then
```

信号が赤の時間

```
          do numb=1,nnt
           x1=x(numb,1)
           x2=x(numb,2)
        if(numb.eq.nnt)then
           x(numb+1,1)=x(1,1)+ll
          endif
           dx=x(numb+1,1)-x(numb,1)

        s=x(numb,1)/ll
        if(((200*s+100-x(numb,1)).le.dx).and.
     a((200*s+100-x(numb,1)).ge.0.5 )) then
           dx=200*s+100-x(numb,1)
        endif
```

信号機までの距離が０．５より長く（急ブレーキを防ぐため信号との距離がほとんどない車は、進ませる）、かつ信号機までの距離が一番短い車は、車間距離の代わりに信号機との距離で速度を変化

```
        if(numb.eq.1)then
         x(numb-1,1)=x(nnt,1)-ll
        endif

           dx2=x(numb,1)-x(numb-1,1)
           call calc_next_step(time,x1,x2,step,x1_next,x2_next,dx,dx2)
           x1=x1_next
           x2=x2_next

            if((dx.le.1.9).or.(x2.le.0.9)) jam=jam+1
```

信号の影響を受けている車を数える

```
           x(numb,1)=x1
           x(numb,2)=x2
          enddo
c--------
        else
```

信号が青の時

```
          do numb=1,nnt
           x1=x(numb,1)
           x2=x(numb,2)
```

```
           if(numb.eq.nnt)then
           x(numb+1,1)=x(1,1)+ll
          endif
           dx=x(numb+1,1)-x(numb,1)
          if(numb.eq.1)then
           x(numb-1,1)=x(nnt,1)-ll
          endif
           dx2=x(numb,1)-x(numb-1,1)
           call calc_next_step(time,x1,x2,step,x1_next,x2_next,dx,dx2)
           x1=x1_next
           x2=x2_next

            if(dx .le. 0) nac=nac+1
            if((dx.le.1.9).or.(x2.le.0.9)) jam=jam+1

           x(numb,1)=x1
           x(numb,2)=x2

          end do
         endif
c-----                            end of car loop

            if(nstep.eq.(nstep/50)*50) then
            istep=istep+1
c------                                keep the car positions
               do icar=1,nnt
              xperiod=x(icar,1)
                do while(xperiod.gt.0)
                 xperiod=xperiod-ll
                enddo

                        xdata(icar,istep)=xperiod+ll
                        vdata(icar,istep)=x(icar,2)
                        ja(istep,1)=jam
              end do
            endif
         end do
c-----------------------------------      end of simulation loop
```

```
            do istep=1,6000
             do icar=1,1
             fstep=istep*3
             dxdata(icar,istep)=xdata(icar+1,istep)-xdata(icar,istep)
             dxdata(nnt,istep)=xdata(1,istep)-xdata(nnt,istep)

           if(dxdata(icar,istep).le.0)then
            dxdata(icar,istep)=dxdata(icar,istep)+ll
           endif

           adata(icar,istep)=(vdata(icar,istep)-vdata(icar,istep-1))/step
c-----------------------------------     print
            write(*,100) fstep,icar,xdata(icar,istep),dxdata
     a(icar,istep),vdata(icar,istep),adata(icar,istep),ja(istep,1)
100   format(i10,3x,i10,3x,f10.3,3x,f10.3,3x,f10.3,3x,f10.3,3x,i10)
信号の影響を受けている車の数も出力

              enddo
           enddo
       stop
       end
c===============================================================
       subroutine calc_next_step(t,x1,x2,step,x1_next,x2_next,dx,dx2)
       implicit none
       real t,x1,x2,step
       real x1_next,x2_next,dx,dx2
       real diff_eq_1,diff_eq_2
       real k1_1,k2_1,k3_1,k4_1,k1_2,k2_2,k3_2,k4_2
        k1_1=step*diff_eq_1(t,x1,x2)
        k1_2=step*diff_eq_2(t,x1,x2,dx,dx2)
        k2_1=step*diff_eq_1(t+0.5*step,x1+0.5*k1_1,x2+0.5*k1_2)
        k2_2=step*diff_eq_2(t+0.5*step,x1+0.5*k1_1,x2+0.5*k1_2,dx,dx2)
        k3_1=step*diff_eq_1(t+0.5*step,x1+0.5*k2_1,x2+0.5*k2_2)
        k3_2=step*diff_eq_2(t+0.5*step,x1+0.5*k2_1,x2+0.5*k2_2,dx,dx2)
        k4_1=step*diff_eq_1(t+step,x1+k3_1,x2+k3_2)
        k4_2=step*diff_eq_2(t+step,x1+k3_1,x2+k3_2,dx,dx2)
        x1_next=x1+(k1_1+2.0*k2_1+2.0*k3_1+k4_1)/6.0
        x2_next=x2+(k1_2+2.0*k2_2+2.0*k3_2+k4_2)/6.0
```

```
return
end

real function diff_eq_1(t,x1,x2)
implicit none
real t,x1,x2
 diff_eq_1=x2
return
end

real function diff_eq_2(t,x1,x2,dx,dx2)
implicit none
real t,x1,x2
real legal,legal2,dx,dx2
 diff_eq_2=1.5*(legal(dx)*legal2(dx2)-x2)
return
end

real function legal(dx)
implicit none
real dx,n
parameter (n=2.0)
legal=tanh(dx-n)+tanh(n)
return
end

real function legal2(dx2)
implicit none
real dx2, n
parameter (n=1.3)
legal2=1.0+(1.0-tanh(dx2-n))/(1.0+tanh(n))/2
return
end
```