PhysicsLab. 2016

計算機班 詳細解説

各プログラム解説

メンバー
羽柴聡一朗、上野智久、桶作愛嬉、亀井健一郎、柴田直幸、周静芳、杉山素直、丹波翼、古川健人、森脇可奈、横島亘

はじめに

ここでは各班員の作成したプログラムの原理、及びその結果などについて詳しい解説を載せています。五月祭の展示で実際に動かしたプログラムの背景にはどんな法則が隠れているのか、是非読んでみて下さい。なお、それぞれのプログラムを作成した班員は表の通りです。（プログラム名の五十音順）

イジング模型	古川健人
宇宙シミュレーション	周静芳、森脇可奈
交通流シミュレーション	羽柴聡一朗
太陽系シミュレーション	杉山素直
ニューラルネットワーク	上野智久
有限環探索	横島亘
粒子法を用いた大規模計算	桶作愛嬉、亀井健一郎、柴田直幸

目次

はじめに i

第 1 章 宇宙シミュレーション 1
1.1 大規模構造 1
1.2 シミュレーション方法 1
1.3 結果 1

第 2 章 セルオートマトンを用いた交通流シミュレーション 3
2.1 導入 3
2.2 セルオートマトン ルール 184 3
2.3 反応速度による遅延 4
2.4 車間距離に応じた加減速 6
2.5 追越車線のある道路 7
2.6 隘路（ボトルネック）のある道路 9
2.7 まとめ 10

第 3 章 太陽系シミュレーション 11
3.1 多体系 11
3.2 シミュレーション方法 11
3.3 結果 11

第 4 章 ニューラルネットワークで作るオセロ AI 12
4.1 目的 12
4.2 用語解説 12
4.3 方法 12
4.4 実験 (第 1,2 回) 13
4.5 実験 (第 3 回) 14
4.6 実験 (第 4 回) 実験 (第 5 回) 15

第 5 章 有限環探索 16
5.1 代数的構造 16
5.2 数学的構造の探索 17

第 6 章 粒子法を用いた大規模計算 18
6.1 粒子法 18
6.2 並列計算 19

参考文献 21

第 1 章

宇宙シミュレーション

1.1　大規模構造

太陽のような恒星が集まったものを銀河といい、私たちの住む太陽系は天の川銀河の恒星の一つです。観測からは、銀河の分布の仕方には特徴的な構造があることが分かっており、この構造は大規模構造と呼ばれています。大規模構造は、宇宙初期におけるわずかなゆらぎが重力によって成長して出来たと考えることができます。それではそもそも初期におけるゆらぎはどのようにして発生したのでしょうか。その原因を説明する理論の一つとして、インフレーション理論というものがあります。インフレーションとは、宇宙が誕生して間もなく起きたとされる宇宙の急速な膨張のことを言います。今回私たちは、理論に基づいて計算された質量ゆらぎを出発点にして、構造形成の過程をシミュレーションしました。

1.2　シミュレーション方法

今回のシミュレーションでは N 体シミュレーションという手法を採用しました。N 体シミュレーションとは、無限個の粒子からなる質量分布を有限個の粒子に代表させて、それらの相互作用を計算していく方法です。今回の場合は各粒子間の重力を計算します。

$$m\frac{d^2\boldsymbol{x}}{dt^2} = -\sum_{j\neq i}\frac{Gm^2}{|\boldsymbol{x}_i-\boldsymbol{x}_j|^2}\frac{\boldsymbol{x}_i-\boldsymbol{x}_j}{|\boldsymbol{x}_i-\boldsymbol{x}_i|} \tag{1.1}$$

宇宙膨張の効果を入れるために、膨張因子（スケール因子）$a(t)$ を考えます。初期状態 $t=0$ における単位長さ d_0に対して時刻 t における単位長さ $d(t)$ を

$$d(t) = a(t)d_0 \tag{1.2}$$

とすれば、時間に依存する膨張が取り入れられることになります。ただし、膨張を取り入れるのは重力計算のときのみで、実際に表示をするときは空間とともに膨張する座標系における座標を用いました。このため表示上は膨張の様子は見られません。このような座標を共動座標と言います。

1.3　結果

インフレーション理論に基づいた初期条件を用いたときと全くのランダムに粒子を並べたときの結果を以下に示します。

理論に基づいた初期条件で計算した方には筋状の構造が見られるのがわかると思います。これは大規模構造に見られる特徴的な構造でもあります。一方でランダムに並べた方ではこのような構造は見られません。これによってこの初期条件、ひいてはこの理論の妥当性が高まるといえます。

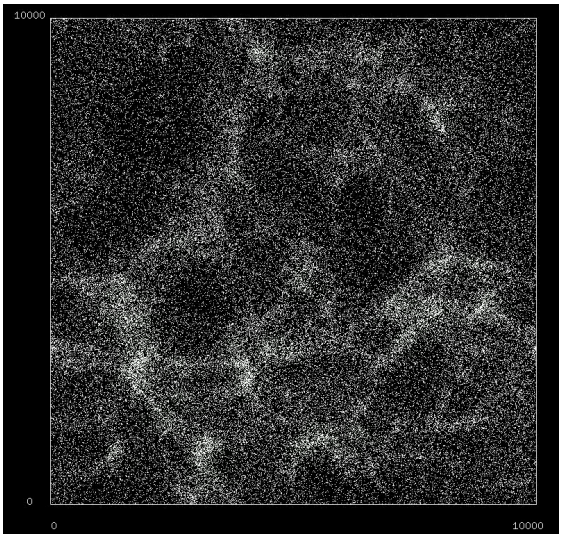

図 1.1　インフレーション理論に基づいた初期条件から始めた場合

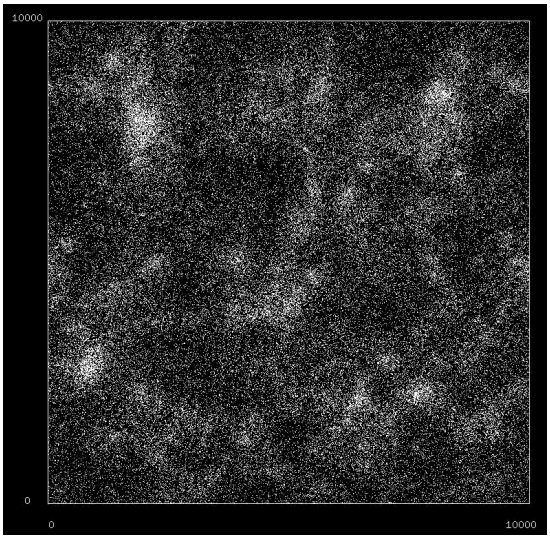

図 1.2　ランダムに並べた状態から始めた場合

第 2 章

セルオートマトンを用いた交通流シミュレーション

2.1　導入

2.1.1　シミュレーションとモデル化

物理学というのは世界を説明しようとする学問です。その際、説明（理論）は簡潔であれば簡潔であるほど良い、美しい理論だとされます。一方、コンピュータシミュレーションというのは物理学の理論を基に世界を再現しようとする試みです。勿論、様々な影響を考慮すればするほど、シミュレーション結果は現実に近付いていくことでしょう。例えば、空気中でのボールの落下を考えると、最も簡単な近似としては質点の自由落下となるでしょうが、球体に対する空気抵抗を考慮し、ボールの形状の影響を考慮し、更には相対論的効果や量子論的効果をも考慮していけばどんどん現実に近付いていく筈です。しかし、それにどれほどの意味があるのでしょうか？恐ろしく複雑なプログラムで現実世界を上手く再現出来たとしても全く面白くないのです。複雑なもので複雑なことが出来るのは当然。簡潔なもので複雑なことが出来てこそ、面白い、美しい物理学と言えるのです。
そして、現実世界をなるべく簡潔に再現する為に欠かせないのがモデル化です。現実世界の裏側にある無数の法則の中から今再現していようとしている現象で特に重要な働きをしているものだけを抜き出してモデル（模型）を作り、箱庭のように現象を再現するのです。このモデル化こそがコンピュータシミュレーションの肝と言えるでしょう。この交通流シミュレーションではこのモデル化という作業に焦点をあて、モデルを改善していくことによってシミュレーション結果が現実に近付いていく様子をご覧に入れたいと思います。

2.1.2　セルオートマトンとは

今回テーマとしたのは自動車の流れ、即ち「交通流」です。この交通流をセルオートマトンという道具を使ってシミュレートしていきます。セルオートマトンとは沢山のマス目（セル）を用意してやって一つ一つのマス目に数字を入れて状態を表し、セルに入った数字を一定の規則に基づいて変化させていくというものです。（今回の例では 0 は車が居ないセル、1 は車が居るセルなど）コンピュータはこういうマス目のようなデジタル（離散的）なものを扱うのが大得意な為、シミュレーションにはとても適していると言えます。なお、このセルオートマトンを使った離散的なモデルとは違い、流体力学の観点から作られた連続的なモデルもありますが、今回は扱わないものとします。気になる方は参考文献 [1] を参考にしてみて下さい。

2.2　セルオートマトン ルール 184

2.2.1　モデルの説明

まず、基礎となる最も簡単なモデルは「前に車が詰まっていたら停車し、そうで無ければ前進する」というもの。具体的には次のような規則に従って車を動かしていきます。

1. 車の 1 つ前のセルに別の車が居なければ、1 つ前のセルに進む（図 2.1 左）
2. 車の 1 つ前のセルに別の車が居るならば、そのセルに留まる（図 2.1 右）

このモデルはセルオートマトン ルール 184 とも呼ばれています。

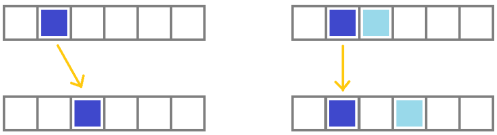

図 2.1　セルオートマトン ルール 184

2.2.2　結果

シミュレーション結果を図 2.2 に、そこから得られた $\rho - \Phi$ 関係式を式 (2.1) に示します。横軸は車の密度 ρ（全セルに占める車の居るセルの割合）、縦軸は交通量 Φ（1 ステップ辺りに基準点を通過する車の数）を表していて、これは交通工学に於いて QK 曲線と呼ばれています。一目で分かる通り、$\rho < 0.5$ では交通量が密度に比例する自由流相となっており、$\rho > 0.5$ では交通量が密度に逆比例する渋滞相となっている事が分かります。こんな単純な規則でも、ある臨界点（この場合は $\rho = 0.5$）を境にして渋滞相が現れました。即ち、渋滞の本質は「ある一定距離以下まで車間距離が詰まると車は減速（停止）する」事にあると言えます。

$$\Phi = \begin{cases} \rho & (\rho < 0.5) \\ 1 - \rho & (\rho > 0.5) \end{cases} \tag{2.1}$$

2.3　反応速度による遅延

2.3.1　モデルの説明

人間というのは刺激を受けてから反応するまでに必ずある一定以上の時間が掛かります。一般的に、随意反射では 0.2 秒程度が反射速度の限界だと言われており、車が発進出来る状態になったことを認識しても実際に発進させるまでには必ず 0.2

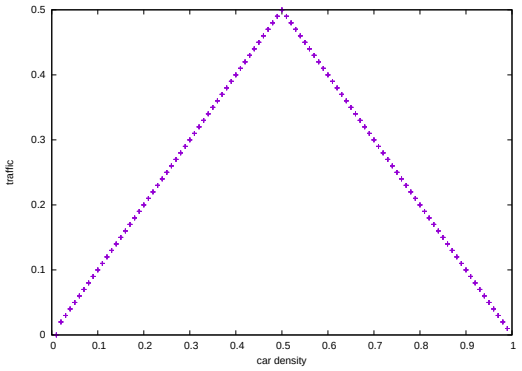

図 2.2　シミュレーション結果 その 1

秒以上の遅れが生じます。まして、誰もがいつも停車中スタートシグナルを待つ F1 レーサーのように神経を研ぎ澄ませている訳ではないので、現実問題として遅延は 1 秒以上になることでしょう。そこで、この遅延をモデルに取り入れてみます。

1. 車が走行しており、かつ車の 1 つ前のセルに別の車が居なければ、1 つ前のセルに進む（図 2.3 左）
2. 車の 1 つ前のセルに別の車が居るならば、そのセルに留まる（図 2.3 中）
3. 停止していた車の 1 つ前のセルが空いた場合、1 ステップおいてから 1 つ前のセルに進む（図 2.3 右）

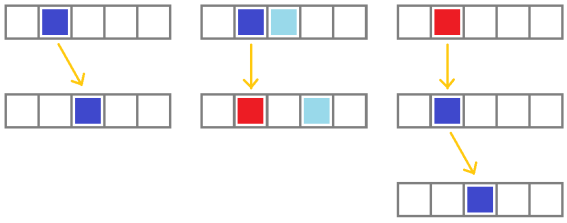

図 2.3　反応速度による遅延を取り入れたモデル

2.3.2　結果

シミュレーション結果を図 2.4 に、そこから得られた $\rho - \Phi$ 関係式を式 (2.2) に示します。今度は $\rho > 0.33$ で渋滞相に転移していることが分かりますが、もう一つ決定的に変わっていることがあります。それが $0.33 < \rho < 0.36$ にまで染み出している自由流相、準安定状態の存在です。このモデルは停止させしなければルール 184 と全く等価なので、偶然皆が停止せずに走り続けられる配置になった場合は $\rho < 0.5$ まで非渋滞相が存在出来るはずです。しかし、誰か 1 台でも停止してしまうとそこから一気に後ろがつかえてしまい、渋滞相へと相転移が生じます。ある意味、過冷却のような状態になっている訳です。この状態は極めて不安定な為、理論的には $\rho < 0.5$ まで伸びるべき渋滞相は、今回は $\rho < 0.36$ までしか観測されませんでした。なお、この準安定状態はプラトーン（小隊）走行とも呼ばれ、高い輸送効率を実現する交通流として注目されています。

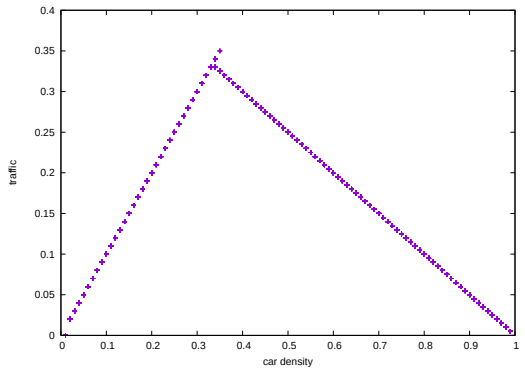

図 2.4　シミュレーション結果 その 2

$$\Phi = \begin{cases} \rho & (\rho < 0.33 + \varepsilon) \\ 0.5(1-\rho) & (\rho > 0.33) \end{cases} \tag{2.2}$$

2.4 車間距離に応じた加減速

2.4.1 モデルの説明

ここまでのモデルでは走行する車の速度は 1 セル/ステップで一定でした。このモデルは発進してすぐに最高速度に達する市街地の路地などでは良く当てはまりますが、幹線道路や高速道路では発進してから最高速度に達するまである程度の時間が掛かり、停車する時も前方の車間距離に応じて徐々に減速していくのが自然です。そこで、車間距離に応じて速度を調整するような規則をモデルに取り入れてみます。停止、発進については反応速度による遅延を取り入れたモデルと同じです。

1. 停止していた車の 1 つ前のセルが空いた場合、1 ステップおいてから速度を 1 にする
2. 速度＞車間距離の場合、速度を車間距離まで落とす（図 2.5 左）
3. 速度＜車間距離かつ速度＜最高速度の場合、1 加速する（図 2.5 右）

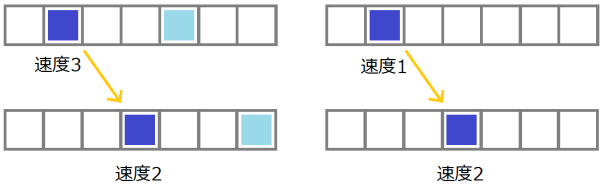

図 2.5 車間距離に応じた加減速を取り入れたモデル

2.4.2 結果

最高速度 v_{max} を 1〜3 にしたシミュレーション結果を図 2.6 に、そこから得られた $\rho - \Phi$ 関係式を式 (2.3) に示します。最高速度が速いほど自由流相での QK 曲線の傾きは大きくなり、かつ渋滞相への臨界密度も小さくなっています。これは走行速度が速いほど早め早めの減速を心掛けるようになり、渋滞が伝播しやすくなる為と考えられます。また、興味深いのが渋滞相の QK 曲線は最高速度を変えても全く変化していないところです。一度渋滞が発生してしまうと皆がノロノロ運転になり、最高速度が意味を成さなくなってしまうのでしょう。また、準安定状態は最高速度が速いほど長く伸びており、最高速度が速いと比較的簡単に準安定状態が実現されることが分かりました。これは高い輸送効率を実現しやすい反面、渋滞相への相転移が起きた際の Φ 変化が急激になる為、重大な事故を引き起こしかねないという面があります。
また、全車のうち割合 p の車は $v_{max} = 1$, 残りの $1 - p$ の車は $v_{max} = 2$ としたシミュレーション結果を図 2.7 に示します。$p = 0.3$ であっても $\rho > 0.08$ で既に $v_{max} = 1$ の QK 曲線と一致しており、遅い車が少しでも居るとその車の速度が全体としての交通流の流速になってしまうことが分かりました。

$$\Phi = \begin{cases} v_{max}\rho & (\rho < (1 + 2v_{max})^{-1} + \varepsilon) \\ 0.5(1 - \rho) & (\rho > (1 + 2v_{max})^{-1}) \end{cases} \tag{2.3}$$

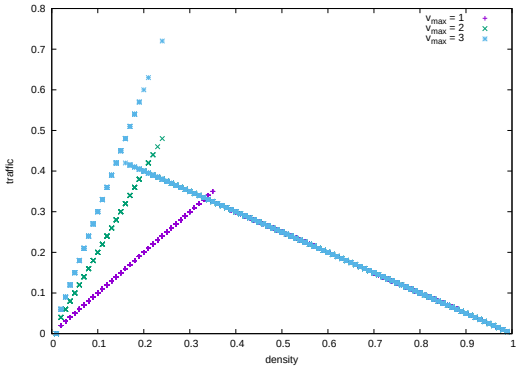

図 2.6　シミュレーション結果 その 3

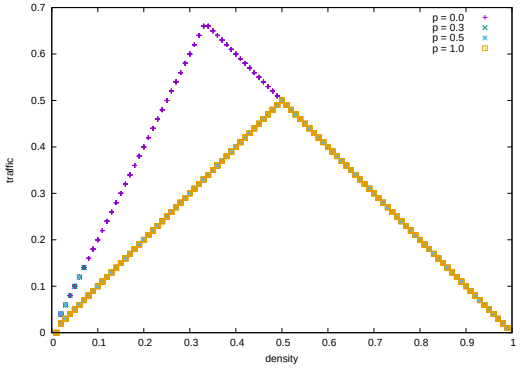

図 2.7　シミュレーション結果 その 4

2.5　追越車線のある道路

2.5.1　モデルの説明

ここまでのモデルでは道路は 1 車線であるとしてきましたが、2.4.2 で見たように 1 車線道路では最も遅い車の速度が全体を律速してしまう為、長距離に渡って走ることが多く、速達性が特に重視される幹線道路や高速道路には追越車線があるのが普通です。そこで、追越車線を導入したモデルを作ってみます。停止、発進、加減速については車間距離に応じた加減速を取り入れたモデルと同じです。原則としては走行車線を走り、已むを得ない場合のみ追越車線を走るという、道路交通法第 20 条第 3 項に則った規則になっています。但し、追越車線に出たり走行車線に戻ったりする際の追突を防ぐ為、各車は次のステップまでに通過し得るセルを前方排除体積として持っており、他の車は次のステップではその前方排除体積に入らないものとします。ちょっと言い方が回りくどいですが、要するに「後ろから自分より速い車が迫ってきていたら車線変更はしない」という当たり前のことを言っているだけです。なお、このモデルについては完全に僕のオリジナルなので、正確性については保証できません。

・走行車線を走っている時は以下の優先順位に基づく

1. 減速せずに走行車線を走り続ける
2. 減速せずに追越車線に出る（図 2.8 左）
3. 減速して走行車線を走り続ける（図 2.8 右）

・追越車線を走っている時は以下の優先順位に基づく

1. 減速せずに走行車線に戻る（図 2.9 左上）
2. 減速せずに追越車線を走り続ける（図 2.9 右上）
3. 減速して走行車線に戻る（図 2.9 左下）
4. 減速して追越車線を走り続ける（図 2.9 右下）

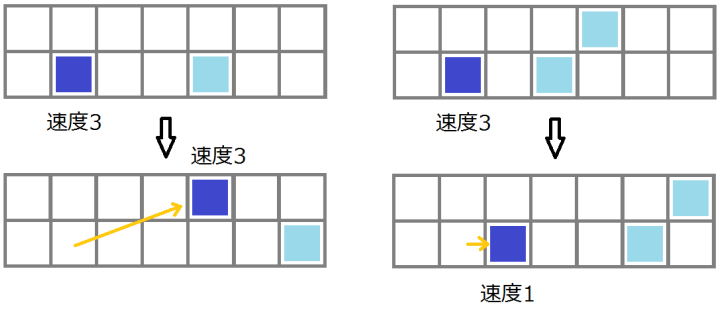

図 2.8　追越車線を取り入れたモデル 1

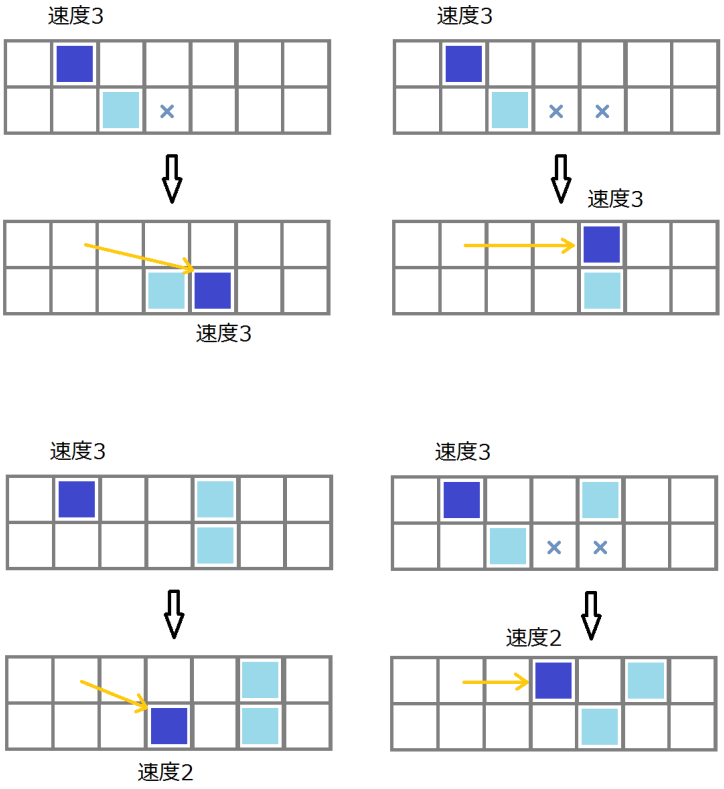

図 2.9　追越車線を取り入れたモデル 2

2.5.2　結果

$v_{max} = 3$ としたシミュレーション結果を図 2.10 に示します。（この図では densityρ は 1 セルあたりの車の数を表しており、単位道路長あたりの車両数はこの 2 倍の値になることに注意。）大きく見ると図 2.6 の $v_{max} = 3$ としたものとほぼ同じ形になっていますが、三つほど決定的な違いが見受けられます。まず、臨界点付近の渋滞相では綺麗な直線には乗ら

ずにある程度の分散が見られるという点です。これは車の初期配置によって走行車線と追越車線に割り振られる車の比が変化することによるものと思われます。次に、図 2.6 では $\rho \simeq 0.14$ で接続していた自由流相と渋滞相が接続しておらず、必ず不連続相転移が生じている点です。1 車線道路に於いては自由流相から渋滞相へ緩やかに移り変わることも可能でしたが、2 車線道路ではある臨界密度（$\rho \simeq 0.2$）を超えるといきなりある程度の長さの渋滞が発生するということになります。いきなりガクンと流れが悪くなるわけで事故の危険があると共に、一度発生した渋滞は 1 車線道路の時と同じ臨界密度 $(1+2v_{max})^{-1} \simeq 0.14$ まで密度を下げなければ解消しないため、輸送効率にも重大な影響を及ぼします。そして最後に、$0.2 < \rho < 0.3$ に於いて現れている第三の相の存在です。Φ が ρ に対して単調減少していることからスムーズな流れでないことは確かですが、真の渋滞相に比べると Φ は 0.4 近く大きい値を示しています。準渋滞相とでも言うべきこの相は自由流相と $\rho \simeq 0.2$ で接続しているため、自由流相からの相転移の際の Φ が余り大きくなく、かつ渋滞相よりも高い密度でも自由流相へと転移できるので、この準渋滞相の発生を迅速に検知して車両密度を調整すれば本格的な渋滞の発生を未然に抑えることが可能であると思われます。

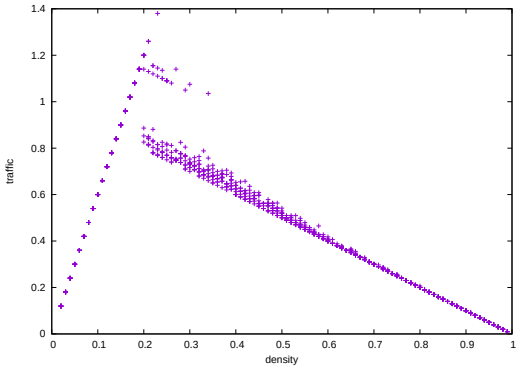

図 2.10　シミュレーション結果 その 5

2.6　隘路（ボトルネック）のある道路

2.6.1　モデルの説明

最後は隘路（ボトルネック）です。規則は 2 車線道路のモデルと全く同じですが、道の一部に動かない車と同様の働きをする障害物を設置し、その部分だけは 1 車線道路になるようにしてみました。いわゆる車線規制を再現したモデルです。また、閉塞感のあるトンネルなども同様の働きをすると言われており、しばしばボトルネック部を先頭にして大渋滞が発生しています。（例：中央自動車道小仏トンネル）

2.6.2　結果

単位道路長あたりの車両数を $p = 0.2, 0.5, 0.9$ に固定し、ボトルネック部の長さを $0 - 100\%$ で変化させたときのシミュレーション結果を図 2.11 に示します。少しでもボトルネックがあるとこの範囲の p では Φ はほぼ同じ値で一定になっており、ボトルネック部の長さがある臨界値を超えると大規模な渋滞が発生するようになって Φ は減少し始めます。これより、ボトルネックが存在する場合は車両密度を調節して下げておいた方が高い輸送効率を長く保てることが分かります。

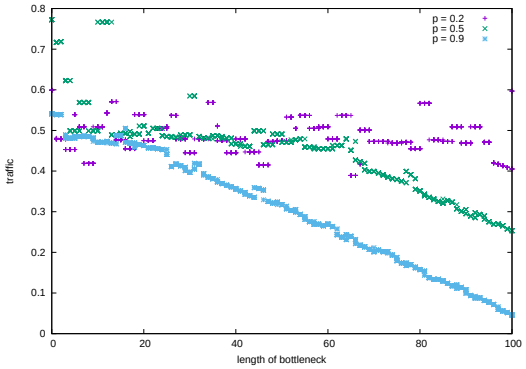

図 2.11　シミュレーション結果 その 6

2.7　まとめ

如何だったでしょうか。モデルを改善するにつれ、より現実に近い交通流になっていく様子が見て取れたことと思います。特に、2 車線道路と隘路のある道路のモデルの結果では、車両密度規制によって輸送効率を高め、事故率を下げる効果があることを示唆しており、実際の高速道路運用に於いても活用できると思います。現実的には高速道路を走っている車両数を制限するとその分一般道が混雑する可能性があり、日本全体としての輸送効率を考えるとそう単純な話ではないでしょうが、試す価値はあるのではないでしょうか。

第 3 章

太陽系シミュレーション

3.1　多体系

相互作用する 2 体扱う問題を 2 体問題、3 体扱う問題を 3 体問題…n 体扱う問題を n 体問題と言います。n 体問題は $n \geq 3$ の時、一般に解析解を持ちません。3 体問題と言われれば簡単に解けそうに思えますが実は特別な場合を除いて解くことはできないのです。このような問題は計算機で数値計算をします。

3.2　シミュレーション方法

今回のシミュレーションは太陽系にある物質を太陽、惑星、月に絞って計算しました。天体の従う方程式は非相対論的重力方程式としました。

$$m_i \frac{d}{dt} \overrightarrow{r_i} = \sum_j -G \frac{m_i m_j}{|\overrightarrow{r_i} - \overrightarrow{r_j}|^2} \tag{3.1}$$

しかし、ただ計算すると言っても計算の方法には様々なものがあり、それぞれに特徴を持っています。今回扱うハミルトン力学系はハミルトニアン（系のエネルギーと思えばいいです）が

$$H(p, q) = T(p) + V(q) \tag{3.2}$$

の形に書ける力学系なので 2 次のシンプレクティック積分法を採用しました。この方法はエネルギーが近似的に（正確には時間の刻み幅の二次の範囲で振動する）保存する特徴を持っています。

3.3　結果

計算した太陽系天体の動きを opengl で見て、各惑星の周期なども計算しました。地球の公転周期を以下に示します。

```
0 year(s)       364 day(s)      21 : 30 : 50
0 year(s)       364 day(s)      21 : 30 : 50
0 year(s)       364 day(s)      21 : 31 : 40
0 year(s)       364 day(s)      21 : 31 : 40
0 year(s)       364 day(s)      21 : 31 : 40
0 year(s)       364 day(s)      21 : 30 : 50
0 year(s)       364 day(s)      21 : 29 : 10
0 year(s)       364 day(s)      21 : 29 : 10
0 year(s)       364 day(s)      21 : 30 : 50
0 year(s)       364 day(s)      21 : 31 : 40
```

図 3.1　地球の公転周期

1 年＝ 365 日、1 日＝ 24 時間、1 時間＝ 60 分、1 分＝ 60 秒としています。よく知っている値になっています。また、もっと長い時間で計算すると公転周期が周期的に変動していました。次のプログラミングでは太陽質量の減少を考慮したモデルや相対論的な効果を取り入れて彗星を飛ばしてみたいと考えています。

第 4 章

ニューラルネットワークで作るオセロ AI

4.1　目的

ニューラルネットワークと遺伝的アルゴリズムを利用して可能な限り強いオセロ AI を作ります。

4.2　用語解説

短い解説にさせていただきます。これらのワードで web 検索すれば多くの情報が手に入ります。

4.2.1　ニューラルネットワーク (NN)

行列演算と非線形関数の作用を組み合わせ、行列の要素をうまく決定することで機械学習やパターン認識が可能となる。

4.2.2　誤差逆伝播法

ニューラルネットワークの行列要素についてのニュートン法のこと。

4.2.3　遺伝的アルゴリズム (GA)

パラメータを変えて環境との適応度を計算し、良いものを生き残らせるシミュレーション。

4.2.4　アルファ・ベータ法

自分は自分にとって最も有利な手、相手は自分にとって最も不利な手を指すと仮定してある程度無駄を省いて局面を総当たりで検索すること。

4.3　方法

4.3.1　基本的な方針

一様乱数を初期値とする NN に後述の非 NNAI を教師として学習を行い、初期乱数を変えて 6 つの AI の雛形を作ります。それらの「子供」を作り AI の数を 9 個にして対局を行って、「強い」ものを生き残らせることを繰り返して世代交代を行います。

表 4.1　マス毎の基本値 (対称性により明らかな場所は省略します)

2000	-1000	-100	-100				
	-4000	-300	-400				
		-200	-300				
			-100				

4.3.2　盤面評価値について

次の一手は、相手が最善の手を指した時に自分が最も有利になる手を選びます。（アルファ・ベータ法）その「有利さ」の指標が盤面評価値です。この盤面評価値の計算にニューラルネットワークを利用することを考えます。但し、この実験では盤面評価値は黒が優勢な場合に正の値をとり、白が優勢な場合に負の値をとるとします。

4.3.3　全ての AI に共通する動作（NN を使う、使わないに関わらずに行われる処理）

- 各 AI は、規定の読む局面数が定義されています。始めは 3 手の深さでアルファ・ベータ法を行いますが、もし評価した局面の数が $\frac{\text{規定の読む局面数}}{5^n}$ 未満だった場合は（3+n）手の探索をやり直します。
- 決着がついたときには評価値が石数の差の定数倍に一致するようにします。これを実現するためには、盤面評価値の陣形部分を各 AI に任せ、石数部分の評価を石数の差の定数倍にしてそれらを対局の進行度に合わせて比をとって考えるということをします。（終盤に行くにつれて各 AI の特徴は薄れ石数のみで判断するようになります）
- 手にランダム要素を入れます。これは、盤面評価値の陣形部分に乱数を足し算することで実現します。（後述）
- 全滅を回避する必要があります。(＊実験 (第 5 回) で実装予定) オセロは序盤に於いては石が少ないほうが概して有利なため、AI の評価も高くなりますが、全滅してしまうと負けてしまうため、その場合だけ評価値を下げることが必要です。（なお、稀なケースとして全滅はしていないものの序盤で双方が手詰まりになることもありますが、それは今は考えないとします）

4.3.4　非 NN の AI について

ニューラルネットワークを利用しない AI が誤差逆伝播法と遺伝的アルゴリズムの適合値の決定の際に必要です。（非 NNAI と略記します）この AI の盤面評価値の決定方法を記します。まず、盤面に数字を対応させます。盤上の全ての石について、黒石ならマス毎の基本値を加算し、白石なら減算します。但し、付近の角に石がある場合は-1000 と-4000 の基本値を 0 とみなします。このように計算された値が非 NNAI の盤面評価値です。

4.4　実験 (第 1,2 回)

バグと不手際によって意味をなしていないので割愛します。

4.5 実験 (第 3 回)

4.5.1 セットアップ

まず、4 層 (ノード数 64,16,4,1) の NN を用意し、学習率 0.01、活性化関数を tanh、行列要素の初期値をそれぞれ $[-\sqrt{\frac{6}{\text{前後のノード数の和}}}, \sqrt{\frac{6}{\text{前後のノード数の和}}}]$ の一様乱数にします [2] この NN に非 NNAI を教師として誤差逆伝播法を行いました。但し、教師信号は tanh の値域 (-1,1) に含まれる必要があるため、非 NNAI の評価値を規格化因子=32000 で割ったもので学習させて使用する際に規格化因子を掛けるということを行いました。評価する盤面はすべてのマスに黒石、白石、空白を $\frac{1}{3}$ の確率で配置したものを使用しました。(当然、実際には出現しない局面が多く含まれます。) 盤面の回転対称性 (後述) を考慮して、5120 通りの盤面*向き 4 方向=20480 回の学習を行いました。これを初期乱数を変えて 6 回行い、6 つの NN を作ります。これらに対して、以下の順を繰り返します。まず、6 つの NN の AI をスロット 0〜5 と呼ぶこととします。次にスロット 0、1 の子供をスロット 6 に、スロット 2、3 の子供をスロット 7 に、スロット 4、5 の子供をスロット 8 に作ります。但し子供は、全ての行列要素について両親の対応する部分の s:(1-s) の内分を取った値とします (s は (0,1) の一様乱数) さらに、親子の全てに対して 0.03 の確率で突然変異を起こします。突然変異が起こった場合は全行列要素について 0.05 の確率で値に 0.3*r を加算します (r は (-1,1) の一様乱数)
さて、こうして生成した 9 つの AI について非 NNAI と先後を変えて 5 セット、つまり 10 回の対局を行い AI が獲得した石数の合計 (に 1 を足したもの) を「適応度」と定義します。9 つの AI のうち、適応度が最高なもの (複数ある場合はランダムに一つ選ぶ) は無条件で生存とし (エリート選抜)、残りの 5 枠については適応度に比例する確率で当選するルーレット選抜を行い、6 つの AI を次世代に引き継ぐ手続きを行いました。以上を 90 世代繰り返しました。

4.5.2 結果

適応度は徐々に上がっていきましたが、非 NNAI より強くはなりませんでした。

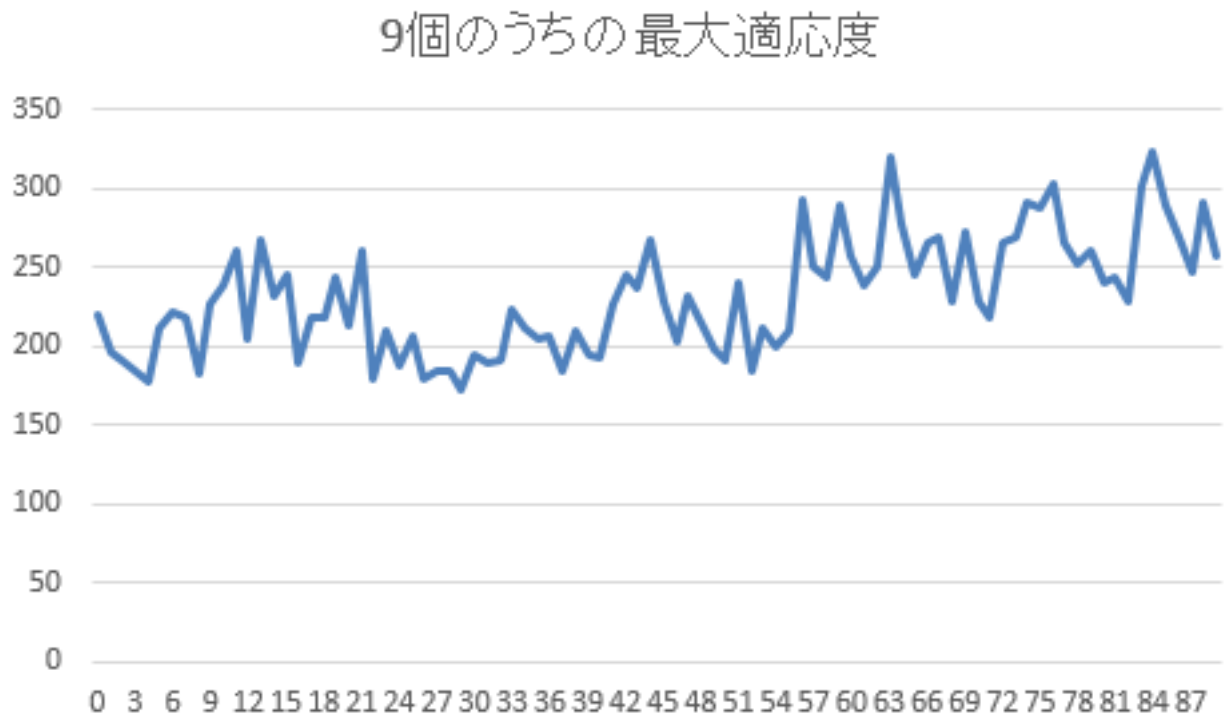

図 4.1 適応度

4.5.3 考察・改善点

- AI の思考ルーチンに盤面評価値の回転、反転対称性、白黒入れ替え反対称性を考慮することが考えられます。
- 獲得石数を重みとするルーレットでは適応度 150 と 200 の重み差が不十分でした。試合の結果自体に大きなランダム要素を含むので、全てエリート選抜にすれば早く学習が進むと考えました。

4.6 実験 (第 4 回) 実験 (第 5 回)

会場で説明しますので是非お越しください。

第 5 章

有限環探索

5.1　代数的構造

1 足す 1 は 2 です。3 掛ける 2 は 6 です。このように僕たちは当たり前に足し算、掛け算をしていますが、それは中学までの話。高校では行列、整数の合同式など、今までとは異なる代数的構造が出てきます。
例えば $1+1=0$ だったら。例えば $3\times 2=2$ だったら。数が有限個しかなかったら。ただ、滅茶苦茶に演算を定めていたのでは何も面白くありません。あるルールを満たした代数的構造のことを群、環、体などと呼ばれています。

5.1.1　群

群は 1 つの演算を持っています。演算が以下の 3 つの条件を満たすと群になります。
(1) 結合則が成り立つ。

$$a+(b+c)=(a+b)+c \tag{5.1}$$

(2) 単位元 0 が存在する。

$$a+0=0+a=a \tag{5.2}$$

(3) 全ての a に対して逆元 x が存在する。

$$a+x=x+a=0 \tag{5.3}$$

例えば、0 と 1 だけからなる集合において表 1 のような演算は群となります。
群は基本的な構造で、普通の数の足し算、行列の足し算など、ほとんどの代数的構造が群となっています。

表 5.1　群

+	0	1
0	0	1
1	1	0

群の演算表は、同じ行、列に同じ数字が入ることはないので、ナンプレにルールが追加されたものと思うことができます。

5.1.2　環

環は 2 つの演算 $+,\times$ を持っています。演算が以下の 3 つの条件を満たすと群になります。
(1)$+$ に関して可換群をなす。

(2)× に関して結合則が成り立つ。

$$a(bc) = (ab)c \tag{5.4}$$

(3) 分配則が成り立つ。

$$a(b+c) = ab + ac(a+b)c = ac + bc \tag{5.5}$$

例えば、上の表に加えて表 2 のような演算があると環となります。

表 5.2　環

×	0	1
0	0	0
1	0	1

5.1.3　ナンプレ

ナンバープレースというパズルで、9 × 9 のものが一般的です。縦、横の各列に同じ数字が入ってはいけないというものです。
結合律などは成り立ちませんが、3 行 4 列目の場所に 2 がある、ということを演算だと思うと、ナンプレも数学的構造と思うことができます。

5.2　数学的構造の探索

上のような数学的構造を計算機で探索します。ナンプレを例として説明します。
1 行 2 列目のマスを (1,2) などと書くことにします。
表 3 のようなナンプレがあったとします。まず、空きマスの候補は 1,2,3,4 となっています。次に、埋まっているマスと同

表 5.3　ナンプレ 1

2			
	3		
			4

表 5.4　ナンプレ 2

2	1	4	3
	3		
	2		4
	4		

表 5.5　ナンプレ 3

2	1	4	3
1	3		
3	2		4
?	4		

じ行、列のマスの候補を消していきます。例えば、(1,2) は、2 と 3 の候補が消え、候補は 1,4 となります。
しかし、このあとに埋まるマスがありません。そこで、候補の中から 1 つずつ仮定していきます。ここでは (1,2) を 1 と仮定します。すると候補が 1 つだけのマスが他にも出てくるので埋めます。(表 4)
次に、(2,1) を 1 と仮定します。すると (3,1) の候補は 3 だけになります。しかし、(4,1) の候補が無くなってしまいました。(表 5)
つまり、(2,1) が 1 であるという仮定は間違っていたことになります。この時点でこの探索はやめ、次の仮定に移ります。
このように、仮定を重ねていき矛盾が出たらすぐに前の仮定に戻る探索法をバックトラッキング法といいます。この方法を使って群、環、ナンプレの探索を行います。

第 6 章

粒子法を用いた大規模計算

京コンピュータがある理化学研究所 計算科学研究機構 (AICS) で、昨年の夏、及び今年の春にスパコンについてのスクールがあり、計算機班のメンバー 3 人で勉強に行きました。そして、今年の計算機班の展示に伴い、東大にあるスパコン FX10 を使わせていただけることになりました。今回の展示では、そのスパコンを用いて「粒子法」と呼ばれる流体シミュレーション手法で「ミルククラウン現象」を解析しました。この粒子法と呼ばれるシミュレーション手法とスーパーコンピュータで行われている「並列計算」について、少し詳しめの説明をさせていただきたいと思います。

6.1 粒子法

6.1.1 流体の運動方程式

流体 (厳密には正しくないが、大雑把に言ってしまえば液体や気体のこと) の運動は、ナビエ=ストークス方程式 (Navier-Stokes equations) と呼ばれる以下の方程式で表されます。

$$\frac{\partial \boldsymbol{v}}{\partial t} + (\boldsymbol{v} \cdot \nabla)\boldsymbol{v} = -\frac{1}{\rho}\nabla p + \nu \Delta \boldsymbol{v} + \boldsymbol{f} \tag{6.1}$$

$$\nabla \cdot \boldsymbol{v} = 0 \tag{6.2}$$

ここで、$\boldsymbol{v}$ は速度、ρ は密度、p は圧力、ν は動粘性係数 (どれくらいネバネバしているかを示す値)、$\boldsymbol{f}$ は外力 (重力など) です。この式自体は 19 世紀前半には既に知られているような式です。そんな昔から分かっているような式をなぜわざわざ取り上げるかというと、この方程式を厳密に解くことは大変難しいからです。その難しさはというと、物理の方程式にも関わらずアメリカのクレイ数学研究所のミレニアム懸賞問題 (リーマン予想やポアンカレ予想という名前は聞いたことがあるという方も多いのではないかと思います) になっていて、この方程式を厳密に解けば[1)]100 万ドルがもらえるほどです。100 万ドルを目指して頑張っている研究者の方も幾人かはいらっしゃると思いますが、それがなされるまで何もしないというわけにもいかない[2)]ので、通常は数値解析によって近似的に解を求めます。

6.1.2 シミュレーションの流れ

物理の問題をコンピュータシミュレーションする場合、一般には以下のような流れを踏むことになります。

1. 対象とする物理をモデル化する。
2. そのモデルの支配方程式を決定する。
3. 適切に空間、時間を離散化する。
4. 上記のことを基にプログラムを書く。

---

1) 実際の出題は、「『3 次元空間と時間の中で、初期速度を与えると、ナビエ=ストークス方程式の解となる速度ベクトル場 $\boldsymbol{v}$ と圧力スカラー場 p が存在して、双方とも滑らかで大域的に定義される。』という主張を証明、もしくは反例を挙げよ。」というもの。

2) 飛行機の設計や津波被害のシミュレーションなど、流体の関わる問題は数多くあります。

「モデル化」とは、実際の物理的要素をすべて取り入れるのではなく、注目したい部分だけを取り出す[3]ということです。この時、いかに必要な部分だけを取り出すことができるかということが極めて重要です。「離散化」とは、簡単に言うと、コンピュータが計算できるような形に式を近似することです。例えば、$\frac{dx}{dt}$ という微分で表された式を $\frac{\Delta x}{\Delta t}$ (Δx、Δt は数字) という分数として扱う、といったことを指します。このような流れを踏んでコンピュータが計算できるようなステップに変換したあと、それに応じたプログラムを書くことになります。

6.1.3 粒子法

数値シミュレーションでナビエ=ストークス方程式を解く方法としては、「有限要素法」や「有限体積法」に代表される「格子法」が従来用いられてきましたが、今回の展示では「粒子法」と呼ばれる新しい計算手法を用いたシミュレーションを行いました。この小節では、粒子法と格子法の違いや粒子法を用いるメリットについて説明したいと思います。

従来の格子法では、流体の物理量を「格子」上に配置し、その格子自体が動くことはありません。格子の中に格納される数値が変化していくのです。一方、粒子法では、流体を「粒子」に粗視化[4]して、その粒子がそれぞれ動くという描像をします。

では、粒子法は従来の格子法と比してどういった点が優れているのでしょうか。それを説明するために、「ラグランジュ記述」と「オイラー記述」と呼ばれる 2 つの流体の記述法について簡単に説明します。「ラグランジュ記述」とは、流れに対して流れとともに視点が移動する記述です。一方、「オイラー記述」とは、空間に固定された視点による記述です。このように書くと難しく聞こえるかもしれませんが、例えば、「雲が西から東に流れていくのを追いかける」のがラグランジュ記述で、「今日の東京の天気は、朝は晴れだが昼頃から雨が降る」というのがオイラー記述だと思っていただければいいと思います。ラグランジュ記述とオイラー記述の定義から、粒子法はラグランジュ記述、格子法はオイラー記述であるということがお分かりいただけるでしょう。実は、式 (6.1) はオイラー記述での方程式であり、ラグランジュ記述では、「ラグランジュ微分」というのを用いたより簡単な形

$$\frac{D\boldsymbol{v}}{Dt} = -\frac{1}{\rho}\nabla p + \nu\Delta\boldsymbol{v} + \boldsymbol{f} \tag{6.3}$$

で記述されるのです[5]。式 (6.1) と式 (6.3) を見比べてみると、式 (6.1) の $(\boldsymbol{v}\cdot\nabla)\boldsymbol{v}$ という項 (対流項と呼ばれます) が式 (6.3) にはあらわには出ていないことが分かります。実は、対流項を精度よく計算することは大変難しく、**対流項をあらわには含まない形で問題を解ける**というのが粒子法の大きなメリットなのです。

6.1.4 ナビエ=ストークス方程式の離散化

前小節では、粒子法を用いるとナビエ=ストークス方程式がより簡単な式 (6.3) の形になることを見ました。後は式 (6.2)、(6.3) を使ってシミュレーションすればいいのですが、ここで新たな問題に直面します。コンピュータができる計算は四則演算くらいであり、∇ や Δ といった偏微分と呼ばれる演算は基本的にはコンピュータで扱うことができないのです。そこで、偏微分を何らかの方法で四則演算に変換する必要があるのです。このような操作を「離散化」と呼びます。ナビエ=ストークス方程式の離散化の詳細は大変複雑なので具体的な導出はここでは省略させてもらいますが、興味のある方は [3] などをご覧ください。

6.2 並列計算

粒子法を用いてより精度の高いシミュレーションをしようと思うと、水の粗視化をより細かくする、すなわち、粒子数を増やさなければなりません。しかし、あまりにも粒子数が増えるとパソコンで計算するのは限界がきます。そのような大規

---

[3] 「そんなことをして厳密に解こうとしないのは甘えではないか。」と思われるかもしれませんが、厳密に問題を解こうとすることは極めて困難で、それは量子力学を扱うような場合には特に重大な問題となります。

[4] ここでいう「粒子」とは計算のための仮想的な考え方であり、分子動力学的に実在する分子を計算するのではなく、流体をある大きさの塊に離散化する、という意味です。

[5] $\frac{D\boldsymbol{v}}{Dt}$ は、速度 $\boldsymbol{v}$ の引数を $(x(t), y(t), z(t), t)$ として、$\boldsymbol{v}(x(t), y(t), z(t), t)$ を t で微分したものと言え、そうすると、確かに式 (6.1) と式 (6.3) は同じ式になることが分かります。

模な計算をしたい時に登場するのがスーパーコンピュータです。スパコンについて詳しくは知らないという方でも、事業仕分けで一躍有名になった「京コンピュータ」はご存知かと思います。この章では、スパコン上で行われている「並列計算」という手法について解説します。

あなたは、次のような足し算、引き算をする必要があったとします。

$$31 + 62 - 5 - 75 + \ldots$$

これが 4、5 個の計算なら簡単にできるでしょう。10 個くらいでも (やる気にさえなれば) できるはずです。しかし、これが 100 個や、もっと増えて 1 万個になったとすると、とても計算する気にはなりません… ところが、もし一緒に計算してくれる人が 1000 人見つかったとするとどうでしょう？一人一人は 10 個の計算をするだけでいいので、一人で 1 万個の計算をするよりは随分簡単になりました。これがスパコンで行われている「並列計算」という手法 (の一例) です。つまり、**一人で解くことは大変困難でも、問題をより小さい問題に分割して、複数人で協力して別々に解く**のです。

並列計算において重要なことが何点かあります。1 点目は「通信」です。先ほどの足し算、引き算の例を考えます。この計算は 1000 人が別々に計算して終わりではありません。1000 人の計算結果を集めて、次の問題 (この例では 1000 個の足し算、引き算) を解かなければなりません。このようにお互いの計算結果を伝え合うことを「通信」と呼んでいます。この通信の時間を短くすることは、効率の良い方法を考えて計算時間を短くすることと同じくらい重要なことです。足し算、引き算の例で言うと、始めの 2 個の計算が終わっただけで「93 になったよ。」といちいち連絡されても、その途中結果を使わないのであれば、それは不要な連絡であり、あなたは (10 個全部計算し終わってから教えて…) と思うでしょう。2 点目は「タスクの振り分け」です。再び足し算、引き算の問題で、A 君は 1 桁の計算ばかり、一方、B 君は 10 桁の計算ばかりだとします。すると B 君は A 君よりもずっと計算時間が必要になってしまうでしょう。全員の計算が終わるまで次の計算が始められないような場合には、一番計算時間のかかる人を待たなければなりませんから、「ある人はとても簡単な計算なのに、別の人はとても難しい計算」という状況は極めて非効率です。裏を返せば、「どの人も同程度の計算時間になるように仕事を振り分ける」方が効率的なのです。この振り分けをいかに上手くできるかが計算時間に大きく影響するのです。

参考文献

[1] 杉原正顯、高安美佐子、和泉潔、佐々木顯、杉山雄規 (2012):『岩波講座 計算科学 6–計算と社会』岩波書店.
[2] Theano で Deep Learning ¡2¿ : 多層パーセプトロン
http://sinhrks.hatenablog.com/entry/2014/11/30/085119
[3] 越塚誠一，柴田和也，室谷浩平，「粒子法入門」(丸善書店, 2015)