重力多体系の数値計算

牧野淳一郎
東京大学理学系研究科天文学専攻

2001 年 2 月 24 日

目 次

1　はじめに

本稿では、基本的に重力で相互作用する粒子（質点）が多数集まって出来ている系、すなわち自己重力多体系について、それがどういうものかということと、それがどのように研究されるか、具体的には、自己重力多体系のコンピュータシミュレーションがどのようなものであり、その方法が自己重力系の物理自体にどのように適応しているかということを簡単にまとめたい。

本稿が『物性研究』の読者にどのように役に立つのか、筆者にもよくわからないが、物理、計算法にはある程度の共通性があり、応用できる面もあると思う。

2　自己重力多体系とは？

自己重力多体系は、基本的には銀河、星団といった多数の恒星が集まって出来ているものである。現実の天文学としては、恒星は有限の大きさを持ち、互いに衝突したり、衝突しないまでも半径程度の距離まで近付いたりし、また進化の結果、超新星爆発で質量の大半を吹き飛ばしたりする。そのような現実的な効果についてはまた後の方で簡単に扱うことにして、とりあえずは古典的な質点から出来た系を考える。

なお、宇宙というと相対論を使うのかという話になるが、ほとんどの対象では相対論的効果は無視出来る。相対論的効果が重要かどうかは基本的には系の中の恒星の運動速度によるが、典型的な銀河ではこれは 200km/s、球状星団や散開星団では 10km/s と光速の 1/1000 以下である。力学進化という観点から重要なのは重力波の放出によるエネルギー損失であるが、この損失率は速度の 10 乗に反比例し極めてちいさくなる。

つまり、基本的な問題は、古典的なニュートン重力法則に従ってお互いを引き合う質点粒子が古典力学の運動方程式に従って運動するような系はどのように時間発展するかということになる。

これはスケールが非常に違うとはいえ分子が多数集まってできている普通のバルクな物質と変わるところはない。従って、普通の統計力学で記述できそうな気がする。他方、重力の他の相互作用とは違った特別な性質として、$1/r$ ポテンシャルで表現される遠距離力でありしかも引力だけであるということがある。このために、通常の物質とは違った振舞いが出てくることになり、またその違った振舞いが通常の MD（分子動力学）シミュレーションとは違った計算法を要請する面もある。

これに対し、MD にしても重力多体系にしても、実際の計算において計算時間のほとんどを占めるのは粒子間相互作用の評価であり、そこをどのように計算精度を保ちつつ高速化するかというのは共通である。

この場は『物性研究』ということなので、あまり重力多体系固有の事情には深入りしないほうがいいのかもしれないが、とりあえずそのあたりを概観してから計算法の話に進むことにしたい。

以下、この節から 4 節までで重力多体系の物理を概観し、5 節以降で数値計算法を見ていく

ことにする。4 節までの内容の半分位は2000 年の物性物理夏の学校の講義ノートと重なっているが、一応このノートはこれ一つで閉じたものにしたいのであえてそういう形をとらせていただいた。

2.1 「統計力学的」アプローチ

ここではまず、統計力学的な扱いという方向から考えてみる。現実の系では星は一つ一つ個性があり、例えば質量も違うわけだが、まずはそういうことは深く考えないで星はみな同じ質量 m を持ち、それが N 個あるということにする。

自己重力系は、そのような N 個の星（以下、「粒子」と書くこともある）が、自分自身が作る重力場の中で運動しているような系である。

さて、統計力学を適用しようというわけだが、すぐに困るのは、実は自己重力系には熱平衡状態がない、つまり、エントロピーが極大値をとる分布というものが存在しないということである。

これはいろんな方法で示すことができるが、例えば以下のようないい方ができる。

熱平衡であるためには、古典統計なので速度分布関数はマックスウェル・ボルツマンでないといけない。しかし、これは不可能である。というのは、マックスウェル分布では速度分布は速度無限大まで広がる（指数関数的にしか落ちない）テイルをもつが、自己重力系では、エネルギーがある値（具体的には、無限遠でのポテンシャルエネルギーの値。普通はこれをゼロ点にとる）以上のものは系から脱出して無限遠に消えていってしまうからである。

この一つの例が 3 体問題である。粒子 3 個を適当な初期状態において、全系のエネルギー（重力のポテンシャルエネルギーと各粒子の運動エネルギーの和）が負になるようにしておくと、しばらくの間は 3 つがそれぞれお互いの回りを動くような状態が続く。しかし、ほとんどすべての場合にこの状態は不安定であり、粒子のうちの 2 つが強く結合した状態になり 3 つめがその反作用ではね飛ばされるという現象が起きる。こうなると、はね飛ばされた粒子はもちろんもうなにもしないし、2 体の系というのはケプラー軌道でこれは無限に安定なので、これは一つの「最終状態」ではある。

もっと粒子数が多い系でも、本質的には似たようなことが起きる。つまり、そのなかでの粒子同士の散乱の結果、高エネルギーの粒子が作られるとそれは系から逃げていってしまうのである。

2.2 自己重力系の熱平衡状態

そういうわけで、統計力学から極めて一般的にいえることは、「自己重力系は、十分長い時間が立てば蒸発してしまう」ということである。しかし、これは確かにその通りではあるものの、あまりものの役には立たない。ほとんどの系で、熱力学的な進化のみによって蒸発が起きるタイムスケールは宇宙年齢よりもはるかに長いからである。大体、我々が（これ

を読んでいるあなたが知りたいかどうかとりあえずおいておくとして）知りたいのは、銀河や星団ではどのようなメカニズムでその形やその中の星の分布が決まっているかということであるから、「いつかはなくなる」といわれてもあんまり嬉しくない。

というわけで、統計力学的な扱いをもう少し意味があるところまで進めようと思うと、2 つの方法が考えられる（他もあるかもしれないけど、あまり調べられていないと思う）。一つは、系の次元を落すことである。粒子が無限遠に逃げてしまうという問題は、空間を 1 次元にすれば回避できる。というのは、1 次元での重力の大きさは距離に依存しない定数であるので、無限遠ではポテンシャルも無限大になるからである。

そういうわけで、一次元系を調べようという話は昔からいろいろある [Tan87, TGK96, TG00] が、とりあえずこのノートではこれについては触れない。というのは、空間を 1 次元化したことで、熱力学的不安定とか、その結果散逸をともなう構造形成が起きることといった自己重力系の興味深い性質のほとんどが失われてしまうからである。そういうわけで、一次元重力系はそれ自体として興味深い対象ではあるが、あまり現実の天体とは関係がない。

もう一つの方法は、壁をつけてしまうことということになる。簡単のために壁は球対称とする。こうしておくと、平衡状態があれば球対称なので算数が簡単になって具合がよろしい。

2.3 無衝突ボルツマン方程式

平衡状態は何かとかいう議論をする前に、支配方程式を決めないと話にならない。そこで、無衝突ボルツマン方程式を導入しておく。いま、粒子数が無限に多い極限を考えると、位相空間での（一体）分布関数 $f(\boldsymbol{x}, \boldsymbol{v})$ は以下の無衝突ボルツマン方程式に従う。

$$\frac{\partial f}{\partial t} + \boldsymbol{v} \cdot \nabla f - \nabla \Phi \cdot \frac{\partial f}{\partial \boldsymbol{v}} = 0, \tag{1}$$

ここで Φ は重力ポテンシャルであり以下のポアソン方程式の解として与えられる。

$$\nabla^2 \phi = -4\pi G \rho. \tag{2}$$

ここで、G は重力定数であり、ρ は空間での質量密度

$$\rho = m \int d\boldsymbol{v} f, \tag{3}$$

である。もちろん、これは、重力多体系の運動方程式

$$\frac{d^2 \boldsymbol{x}_i}{dt^2} = -\sum_{j \neq i, 1 \leq j \leq N} G m_j \frac{\boldsymbol{x}_j - \boldsymbol{x}_i}{|\boldsymbol{x}_j - \boldsymbol{x}_i|^3}, \tag{4}$$

で、質量密度 $\rho = nm$ (ここで n は粒子の数密度) を一定に保って $m \rightarrow 0$（従って $N \rightarrow \infty$）の極限をとったものである。なお、ここで$\boldsymbol{x}_i$ と m_i は粒子 i の位置と質量である。

式は通常のボルツマン方程式と同じであるが、特徴的なのは衝突項がないことである。後で説明するが、自己重力多体系の場合、粒子数無限大の極限では重力場が滑らかになって衝突項が消える。

まず、力学平衡という概念を導入しておこう。これは、(衝突項を無視するという近似のもとで)、分布関数が時間的に定常であるということである。衝突項がないので、一体分布関数に対してリュイビルの定理が成り立つ。従って、ラグランジュ的に見て分布関数の値(位相密度)は一定であり、ボルツマンエントロピーは、少なくとも形式的には保存するということに注意して欲しい。つまり、今考えている粒子数無限大の極限では、通常の意味での熱力学的緩和、つまり熱平衡に向かっての進化は起きない。

で、力学平衡に話を戻す。分布関数が時間的に定常とは、もちろんオイラー的に変化しないということである。このための必要十分条件を与えるのが以下のジーンズの定理である。

ジーンズの定理：任意の無衝突ボルツマン方程式の定常解は、運動の積分を通してのみ位相空間座標に依存する。逆に、任意の運動の積分の関数は定常解を与える。

運動の積分の定義も一応与えておく。ポテンシャル Φ のもとで、ある $\boldsymbol{x}, \boldsymbol{v}$ の関数 I が運動の積分であるとは、その上で

$$\frac{d}{dt} I(\boldsymbol{x}, \boldsymbol{v}) = 0, \tag{5}$$

がなり立つことである。つまり、実際にすべての粒子の軌道について、その上でその量が変化しないということである。ちょっと変形すれば

$$\boldsymbol{v} \cdot \nabla I - \nabla \Phi \cdot \frac{\partial I}{\partial \boldsymbol{v}} = 0 \tag{6}$$

これと、上の無衝突ボルツマン方程式を比べてみると、すぐわかるように時間微分が落ちているだけである。

なお、「運動の積分」というときの流儀は2通りあって、一般に運動の保存量のことを「運動の積分」ということもあるが、ここでは位相空間の座標だけの関数であって同時に保存量であるものをさす。具体的には、たとえば1次元調和振動子で「初期の位相」というのは保存量だが運動の積分ではない。これは、時間が入ってくるからである。これに対し、エネルギー $1/v^2 + \Phi$ や、ポテンシャルが球対称 (r だけの関数) の場合の角運動量ベクトル $\mathbf{L} = \boldsymbol{r} \times \boldsymbol{v}$ は運動の積分である。

以下、定理の証明を一応書いておく。ジーンズの定理をいいかえると、分布関数 f が定常であるためには、運動の積分 $I_1, I_2, ..., I_m$ があって $f = f(I_1, I_2, ..., I_m)$ の形で書けることが必要十分ということになる。

証明だが、まず「定常ならば運動の積分で書ける」というほうを考えてみる。これは、f 自体が運動の積分の定義を満たしているので、OK。

逆のほうは、実際に f の全微分を I_k で書き下せば、それぞれが 0 になるということからいえる。

というわけで、これはなかなか強力な定理だが、一般の場合にはそれほど役に立つわけではない。というのは、ポテンシャルを与えた時に一般に運動の積分というのは 5 個あるはずだが、それらをすべて知っているということは普通はないからである。例えば、ポテンシャルに球対称とか軸対称とかいった対称性がない時には、一般にはエネルギー以外の保存量があるとは限らない。分布に球対称とかいろんな制限を付ければある程度なんとかなることになる。例えば、球対称にすれば、エネルギーの他に角運動量ベクトルの 3 成分が

保存する。もう一つ保存量があるが、これは調和振動子やケプラー運動のような軌道が閉じる場合にしか意味がないので、実効的には上の 4 つが保存量である。ここで、ポテンシャルに対応して分布関数も球対称であるとすると、分布関数がエネルギー E と全角運動量 J にだけ依存するということになる。

軸対称ポテンシャルの場合には、対称軸回りの角運動量 J_z は保存量になる。それ以外には保存量がないと信じると、分布関数が E と J_z で書けるということになる。但し、一般には他に保存量がないかどうかは良くわかっていない。(これには第三積分問題という名前がある)。

ジーンズの定理は、重力多体系の特徴のうち無衝突系 (に近いもの) であることから来るものを端的に表現している。すなわち、通常の流体ならば力学平衡に対応する概念は静水圧平衡である。静水圧平衡であって熱平衡ではない系を我々が扱う時の通常の仮定は局所熱平衡、つまり、十分小さなスケールで見れば熱平衡が成り立っているという仮定である。この仮定により、例えば局所的な温度というものを考えたり、熱とかその輸送を考えたりできるようになる。

ところが、重力多体系の力学平衡では与えられるのは系のどこでも共通である分布関数であり、これは熱平衡でなければもちろん熱平衡分布関数とは違っている。つまり、分布関数が本質的に局所的ではないために、熱平衡でない系では必ず局所的にみても分布関数が熱平衡からずれていて、従って局所的な温度等の熱力学的な量は自明には定義できないということになる。

もちろん、だからといって系が熱平衡に向かわないとかそういうことは (残念ながら) 起こらない。が、その進化は、通常の系のように温度差を打ち消すように熱が流れるといった形では (少なくとも原理的には) 表現できない。

では、そのような熱平衡に向かう進化はどのようにしておきるか ?ということが問題だが、それを理解する枠組が 「2 体緩和」ということになる。

2.4 ビリアル定理

2 体緩和の話にいく前に、一つ重要な関係式を書いておく。定常状態のボルツマン方程式から、速度のモーメントをとったり空間座標のモーメントをとったりしてがちゃがちゃ計算すると、以下のビリアル定理が出てくる。

$$2K + W = 0 \tag{7}$$

ここで、 K は系の全運動エネルギー、つまり、空間各点での $v^2/2$ の平均に質量密度をかけて全空間で積分したものである。W は全ポテンシャルエネルギー

$$W = \frac{1}{2}\int\int \rho(\boldsymbol{x})\rho(\boldsymbol{x}')\frac{1}{|\boldsymbol{x}-\boldsymbol{x}'|}d^3\boldsymbol{x}d^3\boldsymbol{x}' \tag{8}$$

である。

今、系の全エネルギーを E とすれば、 $E = K + W$ であるから、

$$E = -K = W/2 \tag{9}$$

ということになる。つまり、定常状態にある自己重力恒星系では、必ず全エネルギーはポテンシャルエネルギーのちょうど半分であり、絶対値が運動エネルギーに等しい。これは球対称とかそういう仮定なしに常に正しい。このために、系のなかでなにか散逸があったり、あるいは系の外にエネルギーを与えたりすると、全エネルギーの絶対値が大きくなり、そのためにポテンシャルと運動エネルギーの両方がかならずその分大きくなる。

ここで運動エネルギーといっているものは多くの場合に恒星のランダム運動、つまり熱運動のエネルギーであり、それが大きくなるということは大雑把にいって温度が上がるということである。つまり、自己重力系には、基本的にエネルギーを失うほど温度が上がるという性質、すなわち「負の比熱」があるわけである。

なお、式 (8) から、系の全質量を M とした時に

$$W = \frac{GM^2}{2R_{\mathrm{vir}}} \tag{10}$$

となるような長さの次元をもつ量 R_{vir} を定義出来る。これをビリアル半径と呼ぶ。R_{vir} は重力ポテンシャルから見た系の特徴的な大きさを与える。

3　2 体緩和とはなにか？

まず、2 体緩和とはいったいどういうものかというところから話を始めることにする。原理的には、これがなにかというのは結構厄介な問題である。

有限粒子数の自己重力多体系を考えると、これは以下のような進化をすると考えられる。まず、最初は力学平衡になかったとすると、とりあえず力学平衡に落ちつく。粒子数が無限大であれば、無限に細かく見れば無限に時間がたっても真の力学平衡に到達するわけではないが、まあ、漸近はしていく。この時、各粒子は与えられたポテンシャルの中を運動するだけになり、それ以上進化することはなくなる。

さて、実際には有限粒子数であるので、そもそも真の力学平衡というものはない。有限の質量をもった各粒子が系の中を運動するに従って、ポテンシャルは必ず変化するからである。この変化によって各粒子の軌道も変化することになる。

それでは、粒子の軌道の変化を、粒子数が有限であることから来る成分とそれ以外に分離することは可能であろうか？系が力学平衡にあるとみなすことができればそれは可能である。つまり、力学平衡にあれば、粒子のエネルギー変化は定義によりすべて粒子数が有限であることによるからである。

が、良く考えると問題なのは、そもそも有限粒子数であるものを力学平衡とみなすとはどういうことかということである。このあたりを考えていると段々混乱してくるので、以下、理想化された状況から順番に考えていくことにしよう。なお、この節の内容は、L. Spitzer の Dynamical Evolution of Globular Clusters[Spi87] にほぼ沿っている。2 体緩和に関しては、重力多体系での議論とプラズマ物理での議論は全くパラレルなものであり、上記の本の内容自体、同じ著者の Physics of Fully Ionized Gases[Spi56] のものとほとんど同じである。

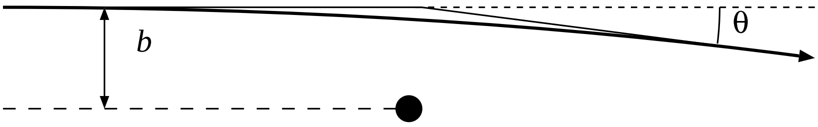

図 1: 2 体散乱

3.1 一様等方な分布

理想化といえば一様等方な分布を仮定することである。例えばマックスウェル分布があって、その中の一つの粒子をとって考えるということをしたいわけだが、これは結構厄介なのでさらに簡単な例を考える。すなわち、速度 0 で空間内に一様（ランダム）に分布した質点を考え、その中を質量 0 のテスト粒子を飛ばして見る。

もちろん、この場合エネルギー交換はないので速度は変わらず、単に散乱されるだけだが、しかし、この例は 2 体緩和のいくつかの重要な性質を示すのですこし詳しく見ていくことにする。分布している質点の質量を m、数密度を n とする。図 1 のように、テスト粒子が一つの粒子から距離（インパクトパラメータ）b を速度 v で通った時に曲がる角度θは、実際にケプラー問題の解析解を使って

$$\begin{aligned} \tan\theta &= \frac{2b}{(b/b_0)^2-1} \\ b_0 &= \frac{Gm}{v^2} \end{aligned} \qquad (11)$$

で与えられる。単位時間当たり、インパクトパラメータが $(b, b+db)$ の範囲にある散乱の回数は $2\pi nvbdb$ である。

さて、散乱の方向はランダムであると思われるので、平均としては（一次の項は）0 になる。しかし、 2 次の項は 0 にならない。これは

$$<\Delta\theta^2>= 2\pi nv\int_0^{b_{max}}\delta\theta^2 bdb \qquad (12)$$

で与えられることになる。

この式から既にいろいろな性質がわかる。が、その前に理論的な困難を解決しておく必要があるであろう。すなわち、この積分は $b\rightarrow\infty$ で発散しているのである。これについてはいくつかの考え方があった。例えば、初めて 2 体緩和の性質を理論的に調べたチャンドラセカール [Cha43] は、以下のように考えた。

平均粒子間距離よりもインパクトパラメータが大きいような散乱は、多体の干渉によって効かなくなるのでそこで積分を打ち切ってよい」

しかし、多体の干渉というようなものが実際にあるかどうかはあきらかではない。もっと素直な解釈は、実際に系にあるすべての粒子と常に同時に相互作用しているのだから、システムサイズくらいまで全部いれる（系が構造を持つ場合はちょっとややこしいが、密度の空間依存も積分のなかに入れて全空間で積分する）というものである。

数値実験の結果などから、後者の解釈すなわち全体が効くというほうが正しいということはかなり昔から大体わかっていた。歴史的には、どちらの解釈が正しいかについてはかなり最近まで論争があって、完全に決着がついたといえるのは 94-5 年頃である。が、現在では後者の解釈が正しいということに疑いの余地はない。

式 (12) から、適当に近似すると

$$< \Delta\theta^2 > \sim Gnv^{-3}m^2\log(R/r_0) \tag{13}$$

となる。ここで R は先に述べたシステムの大きさ、r_0 は「大きく曲がる」ためのインパクトパラメータの値で、$b_0 = GM/v^2$ の程度である。

さて、これからどんなことがわかるかというわけだが、これから、逆に角度変化が 1 の程度になる時間というのを求めてみると、

$$t_\theta \sim \frac{v^3}{Gnm^2\log\Lambda} \tag{14}$$

となる。ここで Λ は上の R/r_0 を単に書き換えただけである。

今、$\log\Lambda$ の質量依存性といったものを無視すると、散乱のタイムスケールは速度の 3 乗、数密度の逆数、質量の 2 乗の逆数に比例するということがわかったことになる。特に、質量密度一定の場合というものを考えてみると、タイムスケールが各粒子の質量に比例するということがわかる。

ある大きさを持った多体系というものを考えてみよう。質量 M、ビリアル半径 R、粒子数 N とすれば、ビリアル定理から¡$v^2 > /2 = GM/R$、力学的なタイムスケール、つまり典型的な速度を持つ粒子が系を横断するのにかかる時間が $t_d = R/v \sim \sqrt{R^3/GM}$ となる。これを使うと上の緩和のタイムスケールは

$$t_\theta \sim \frac{N}{\log N}t_d \tag{15}$$

となる。つまり、力学的なタイムスケールに比べて、2 体緩和のタイムスケールはほぼ N 倍長いということになる。このために粒子数が大きいほど無衝突系に近付くわけである。

3.2 どういうことを考えるかということ :流体との違い

2 体緩和によって、最終的には系が熱力学的に進化するわけであるが、これが普通の流体（ガス）とは本質的に違うものであるということをここで再確認しておこう。

ガスの場合、粒子の平均自由行程はシステムサイズよりもはるかに小さい。液体であれば平均粒子間距離は粒子のサイズ程度であるし、気体であっても通常の状況では考えている

現象の空間スケールに比べて平均自由行程は小さい。ちなみに、非常に希薄な気体とか、あるいは本当に空間スケールの小さい現象では平均自由行程が問題になる。これは例えば超高層での人工衛星の回りの気体の流れとか、あるいは最近の磁気ヘッドの回りの空気の流れとかいったものである。

とにかく、通常のガスの場合、平均自由行程がシステムサイズより小さい。このために、システムサイズよりは小さく平均自由行程よりは大きいような空間スケールを考えると、そのなかでほぼ熱平衡になっていると思っていいことになる。いいかえれば、いわゆる Local thermal equilibrium (LTE) の仮定が使える。こうなると、温度とか圧力とかいった量が近似的（といっても実際上非常に高い近似精度で）に定義でき、そういったマクロな量で系の進化を扱う、特に熱の流れを拡散方程式で書くということが可能になる。

しかし、自己重力質点系では状況が全く異なる。まず、粒子数が無限大の極限では、平均自由行程も無限大であった。つまり、LTE が成り立たないどころか、そもそも熱平衡に向かう（すなわちエントロピーを生成する）ようなメカニズムがなかったわけである。

粒子数が有限の場合も、依然として平均自由行程が長い。つまり、粒子数無限大の時の軌道から、他の粒子との相互作用によって段々ずれていくわけだが、そのずれる典型的なタイムスケールは $Nt_d/\log N$ 程度であった。つまり、流体の場合とは全く逆に、ほとんど衝突はしないで自由運動（他の粒子全体が作るポテンシャルに沿った運動）をしていて、その場が有限の粒子で表現されるための揺らぎがあるので段々軌道が変わっていくということになるわけである。

従って、ローカルな熱平衡を仮定して拡散係数/輸送係数を求めるというのとは逆に、ある一つの粒子が系の中を動き回りながらどういうふうにエネルギー等を変化させていくかという観点で見ていくことになる。

これをすこし別ないい方をすれば、通常の空間のなかでの密度や温度の変化を考える代わりに、また 6 次元位相空間のなかでの分布関数の進化を考えるということに当たる。具体的には、これまで無視してきた「衝突項」というものをちゃんと評価して、どういうものかみてやろうということである。

3.3 バックグラウンドの分布のもとでの有限質量のテスト粒子の振舞い

さて、以下ではバックグラウンドの粒子分布のもとでの一つのテスト粒子の振舞いを考える。3.1 節と違うのは、バックグラウンドも動いていることと、テスト粒子も有限の質量を持つことである。バックグラウンドの粒子は一様に分布するものとし、ある速度分布に従うとする。さらに、バックグラウンドの粒子間の相互作用とかは考えないことにする。これで本当にいいかどうかはちょっと良くわからない問題であるが、まあ、とりあえずやってみることにしよう。

前と同じく、分布している質点の質量を m、数密度を n とする。テスト粒子が一つの粒子から距離（インパクトパラメータ）p を相対速度 $\boldsymbol{V} = \boldsymbol{v}_t - \boldsymbol{v}_f$ で通った時に曲がる角度は、

実際にケプラー問題の解析解を使って

$$\begin{aligned}\tan\theta &= \frac{2p}{(p/p_0)^2-1}\\ p_0 &= \frac{G(m_t+m_f)}{V^2}\end{aligned} \tag{16}$$

で与えられる。ここで、この曲がる角度は相対軌道のものであって、テスト粒子の軌道のものではないということに注意する必要がある。いきなり回りが動いていてしかもテスト粒子が質量をもつというのは面倒になるので、とりあえずテスト粒子は質量を持つが、回りは止まっている場合を考える。この時、一回の散乱での速度変化は以下の式に従う。

$$\Delta v_{\text{垂直}} = \frac{m_f}{m_t+m_f}V\sin\theta = 2V\frac{m_f}{m_t+m_f}\frac{p/p_0}{1+(p/p_0)^2} \tag{17}$$

$$\Delta v_{\text{平行}} = \frac{m_f}{m_t+m_f}V(1-\cos\theta) = -2V\frac{m_f}{m_t+m_f}\frac{1}{1+(p/p_0)^2} \tag{18}$$

$$\tag{19}$$

前の話との違いは、速度変化に係数 $m_f/(m_f+m_t)$ がついていることだけである。これにまた単位時間当たりの衝突回数 $2\pi p n_f V dp$ を掛けて積分するが、$\Delta v_{\text{垂直}}$ については前と同様 1 次の項は落ちる。それ以外についても前と同様に計算出来て

$$< \Delta v^2_{\text{垂直}} > = \frac{2n_f\Gamma}{V} \tag{20}$$

$$< \Delta v_{\text{平行}} > = -\left(1+\frac{m_t}{m_f}\right)\frac{n_f\Gamma}{V^2} \tag{21}$$

$$< \Delta v^2_{\text{平行}} > = \frac{n_f\Gamma}{V\ln\Lambda} \tag{22}$$

ここでΓは

$$\Gamma = 4\pi G^2 m_f^2 \ln\Lambda \tag{23}$$

である。ただし、 leading term でない項は適当に落ちてたりするので注意。

上の式で、$< \Delta v^2_{\text{垂直}} >$の項は前に扱った角度の曲がる項と同じものである。前の話と違うのは、ネットに速度が小さくなる成分がある、すなわち$< \Delta v_{\text{平行}} >$が負で有限の値をもつということである。

これは、 dynamical friction というものである。つまり、回りが止まっているなかを粒子が走っていくと、それが回りを引っ張って動かすので、その分エネルギーを失って段々速度が落ちるわけである。これは、 m が大きい (m_f が小さい) 極限では $m_f n_f$ 、つまり質量密度によっていて、バックグラウンドの粒子の質量に依存しないことに注意してほしい。これに対し、他の項は$m_f^2 n_f$に比例していて、質量密度が同じでも粒子の質量が大きいほうが値が大きくなる。

さて、ここではとりあえず 1 次と 2 次の項を求めたわけだが、それより先の項については考えなくてもいいのだろうか ?ここでは粒子の軌道変化がたくさんの散乱のランダムな重ね合わせで書けるとした。この仮定が正しければ、たくさん散乱を受けた後の速度の分布

は 1 次と 2 次のモーメントで決まるガウス分布になり、従って 3 次より高いモーメントの寄与は考えなくてもいいことになる。

問題はこの仮定が正しいかどうかであるが、実は理論的にはそれほど厳密に正しいわけではない。というのは、インパクトパラメータが例えば p_0 の程度の散乱というのも現実におき、その効果はそれ以外の散乱すべての寄与に比べてせいぜい $\ln\Lambda$ 程度でしか小さくないからである。まあ、しかし、そんなことをいっていても高次の項があっては計算出来ないし、とりあえず $\ln\Lambda$ 程度で小さいということも確かなので、以下高次のモーメントは考えない。

3.4 バックグラウンドが速度分布を持つ場合

いよいよバックグラウンドが動いている場合ということになる。この時でも、相対速度の変化自体は前節に述べたもので正しいが、相対速度にフィールド粒子の速度成分が入ってくるところが違う。

以下、二種類の単位ベクトル系をとって、その上で考える。一つは $(\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3)$ であり、元の空間に固定されている。もう一つは $(\boldsymbol{e}'_1, \boldsymbol{e}'_2, \boldsymbol{e}'_3)$ であり、最初の成分を相対速度 $\boldsymbol{V}$ に平行にとる。従って、後者は相手の粒子によって違うわけである。この 2 つを考えることで、相対速度の変化をもとの静止系でのテスト粒子の変化に焼き直す。

まず、1 次の項は相対速度に平行な成分だけであった。このことから、ある方向の速度変化は

$$< \Delta v_i >=< \boldsymbol{e}_i \cdot \boldsymbol{e}'_1 \Delta v_{平行} >' \tag{24}$$

ということになる。もうすこし精密に書くと、右辺はインパクトパラメータと相手の速度の積分なので、以下のように書けることになる。

$$\begin{aligned} < \Delta v_i > &= \int dp 2\pi p V \int d\boldsymbol{v}_f f(\boldsymbol{v}_f) \Delta v_i & (25) \\ &= -\Gamma(1 + m/m_f) \int \frac{f(\boldsymbol{v}_f)}{V^2} (\boldsymbol{e}_i \cdot \boldsymbol{e}'_1) d\boldsymbol{v}_f & (26) \end{aligned}$$

ここで、$\boldsymbol{v}_f = \boldsymbol{v}_t - \boldsymbol{V}$ はフィールド粒子の速度、f は速度分布関数である。p についての積分を先にやったことに注意して欲しい。この積分は $v_f = 0$ であった時の結果をそのまま使っている。

さて、2 次の項についてであるが、前節で見たように $\boldsymbol{V}$ に平行な成分は小さいので、垂直な成分、すなわち e'_2 と e'_3 の成分を考えればいい。従って、一つの方向からくるフィールド粒子との散乱を考えた時には

$$\begin{aligned} < \Delta v_i \Delta v_j > &= [(\boldsymbol{e}_i \cdot \boldsymbol{e}'_2)\boldsymbol{e}'_2 + (\boldsymbol{e}_i \cdot \boldsymbol{e}'_3)\boldsymbol{e}'_3] \cdot [(\boldsymbol{e}_j \cdot \boldsymbol{e}'_2)\boldsymbol{e}'_2 + (\boldsymbol{e}_j \cdot \boldsymbol{e}'_3)\boldsymbol{e}'_3] < \Delta v^2_{垂直} > /2 \\ &= [(\boldsymbol{e}_i \cdot \boldsymbol{e}'_2)(\boldsymbol{e}_j \cdot \boldsymbol{e}'_2) + (\boldsymbol{e}_i \cdot \boldsymbol{e}'_3)(\boldsymbol{e}_j \cdot \boldsymbol{e}'_3)] < \Delta v^2_{垂直} > /2 \\ &= Q_{ij} < \Delta v^2_{垂直} > /2 & (27) \end{aligned}$$

となる。最後の Q_{ij} は角括弧の中をそう書いただけである。(27) 式にまた分布関数を掛け

て積分すると、結局

$$< \Delta v_i \Delta v_j >= \Gamma \int \frac{f(\boldsymbol{v}_f)}{V} Q_{ij} d\boldsymbol{v}_f \tag{28}$$

ということになる。これで一応必要な 2 次までの係数はすべて書けたわけだが、あまり計算するのに使い易い形ではない。というのは、e'_i とか V とかいったものがまだややこしい形ではいったままであるからである。しかし、もうちょっと簡単な形に書き直せることが知られている。まず、1 次の項だが、

$$\frac{\partial}{\partial v_i}\left(\frac{1}{V}\right) = -\frac{1}{V^2}\frac{\partial V}{\partial V_i}\frac{\partial V_i}{\partial v_i} = \frac{V_i}{V^3} = -\frac{\boldsymbol{e}_i \cdot \boldsymbol{e}'_1}{V^2} \tag{29}$$

という都合のよい関係がある。$\boldsymbol{e}'_1$をそもそも $\boldsymbol{V}$ に平行にとったから上のように出来るわけである。このため、

$$h(\boldsymbol{v}) = \int \frac{f(\boldsymbol{v}_f)}{|\boldsymbol{v} - \boldsymbol{v}_f|} d\boldsymbol{v}_f \tag{30}$$

なる関数 $h(\boldsymbol{v})$ を導入して、

$$< \Delta v_i >= -\Gamma(1 + m/m_f)\frac{\partial h}{\partial v_i} \tag{31}$$

ということになる。

2 次の項についても同様な整理が可能である。Q_{ij}は$\boldsymbol{e}_i$, $\boldsymbol{e}_j$ のe'_2とe'_3によって張られる平面への写像の内積なので、e'_1との内積の分をつけてやれば元の単位ベクトル同士の内積になる。つまり、

$$Q_{ij} = \delta_{ij} - (\boldsymbol{e}_i \cdot \boldsymbol{e}'_1)(\boldsymbol{e}_j \cdot \boldsymbol{e}'_1) = \delta_{ij} - \frac{V_i V_j}{V^2} \tag{32}$$

したがって、

$$\frac{\partial^2 V}{\partial v_i \partial v_j} = \frac{1}{V}\left(\delta_{ij} - \frac{V_i V_j}{V^2}\right) = Q_{ij}/V \tag{33}$$

というわけで、

$$g(v) = \int f(\boldsymbol{v}_f)|\boldsymbol{v} - \boldsymbol{v}_f| d\boldsymbol{v}_f \tag{34}$$

とおけば、

$$< \Delta v_i \Delta v_j >= \Gamma \frac{\partial^2 g}{\partial v_i \partial v_j} \tag{35}$$

3.5 バックグラウンド速度分布が熱平衡の場合

前節では、バックグラウンドの速度分布が任意のものについて、実際に計算可能な式を導いた。ここでは、速度分布が等方的な場合について式をさらに単純化してみる。

速度分布が等方的な場合、h や g の積分を、 $\boldsymbol{v}_f$の絶対値方向と角度方向に分けることができる。角度方向の積分については、 $\boldsymbol{v}$ と $\boldsymbol{v}_f$のなす角度を θとし、$\mu = \cos\theta$ とすれば、球面上での積分が、まず h については

$$\int \frac{d\boldsymbol{v}_f}{|\boldsymbol{v} - \boldsymbol{v}_f|} = 2\pi \int_{-1}^{1} \frac{d\mu}{|v^2 - 2\mu v v_f + v_f^2|^{1/2}} = 4\pi \begin{cases} 1/v, & (v > v_f) \\ 1/v_f, & (v < v_f) \end{cases} \tag{36}$$

となる。これは、球面上に分布する電荷の作るポテンシャルと同じ式である。g についても同様に計算できて

$$\int |\boldsymbol{v}-\boldsymbol{v}_f| d\boldsymbol{v}_f = 2\pi \int_{-1}^{1} |v^2 - 2\mu v v_f + v_f^2|^{1/2} d\mu = \frac{4\pi}{3} \begin{cases} 3v + v_f^2/v, & (v > v_f) \\ 3v_f + v^2/vf, & (v < v_f) \end{cases} \tag{37}$$

となる。これから、

$$\begin{aligned} F_n(v) &= \int_0^v \left(\frac{v_f}{v}\right)^n f(v_f) dv_f \\ E_n(v) &= \int_v^\infty \left(\frac{v_f}{v}\right)^n f(v_f) dv_f \end{aligned} \tag{38}$$

というものを考えると、

$$\begin{aligned} h(v) &= 4\pi v [F_2(v) + E_1(v)] \\ g(v) &= \frac{4\pi v^3}{3} [3F_2(v) + F_4(v) + 3E_3(v) + E_1(v)] \end{aligned} \tag{39}$$

これらから、最終的な結果、すなわち、バックグラウンドが動いている時の、速度に平行な速度変化と垂直なそれを書き下せることになる。それらは、結局、

$$\begin{aligned} < \Delta v_{平行} > &= -4\pi\Gamma \left(1 + \frac{m}{m_f}\right) F_2(v) && (40) \\ < \Delta v^2_{平行} > &= \frac{8\pi\Gamma v}{3} [F_4(v) + E_1(v)] && (41) \\ < \Delta v^2_{垂直} > &= \frac{8\pi\Gamma v}{3} [3F_2(v) - F_4(v) + 2NE_1(v)] && (42) \\ & && (43) \end{aligned}$$

これらから、粒子のエネルギーの変化 ΔE を出すことができる。

$$\Delta E = v\Delta v_{平行} + < \Delta v^2_{平行} > /2 + < \Delta v^2_{垂直} > /2 \tag{44}$$

と書けるので、1 次の項は

$$< \Delta E > = 4\pi\Gamma v \left[E_1(v) - \frac{m}{m_f} F_2(v)\right] \tag{45}$$

となる。2 次の項については、$(v\Delta v_{平行})^2$以外の項は小さいので無視すると

$$< \Delta E^2 > = \frac{8\pi\Gamma v^3}{3} \left[F_4(v) + E_1(v)\right] \tag{46}$$

となる。

さて、速度分布を熱平衡、すなわち

$$f_0(\boldsymbol{v}) = \frac{n_f}{(2\pi\sigma^2)^{3/2}} \exp\left(\frac{-v^2/2}{\sigma^2}\right) \tag{47}$$

とすると、上の係数等を具体的に計算できることになって、その形は

$$\begin{aligned}
< \Delta v_{平行} > &= -4\frac{n_f\Gamma}{\sigma^2}\left(1+\frac{m}{m_f}\right)G(x) && (48)\\
< \Delta v^2_{平行} > &= 2\sqrt{2}\frac{n_f\Gamma}{\sigma}G(x)/x && (49)\\
< \Delta v^2_{垂直} > &= 2\sqrt{2}\frac{n_f\Gamma}{\sigma}\frac{\mathrm{erf}(x)-G(x)}{x} && (50)\\
< \Delta E > &= \sqrt{2}\frac{n_f\Gamma}{\sigma}\left[-\frac{m}{m_f}\mathrm{erf}(x)+\left(1+\frac{m}{m_f}\right)x\mathrm{erf}'(x)\right] && (51)
\end{aligned}$$

ここで erf は誤差関数であり、

$$G(x)=\frac{\mathrm{erf}(x)-x\mathrm{erf}'(x)}{2x^2} \qquad (52)$$

また $x=v_t/(\sqrt{2}\sigma)$ である。

山ほど式はでてきたものの、だからなんなんだという気もする。以下、上の式の意味についてちょっと考えてみる。

まず、速度の 1 次の項を見てみる。これは、速度分布には F_2 だけを通して依存しているということに注目して欲しい。例えば、マックスウェル分布のようなものを考えた時、 v が大きい極限では $F_2\sim 1/(2\pi v^2)$ となるので、回りが止まっているときと同じく速度変化は速度の 2 乗に反比例する。これに対して、v が小さい極限では、f を一定と見なすことが出来るので $F_2\propto v$ となる。

これは、タイムスケールを考えてみると、速度が大きい極限では減速のタイムスケールが v^3 であるのに対し、逆の極限では 一定になるということである。すなわち、非常に速度が大きい粒子が出来てしまうとこれはなかなか減速しない。もちろん、自己重力系の場合には、そのようなものは系のなかに留まるのは困難である。このような粒子 (超熱的粒子) が問題になるのはプラズマの場合である。

速度が小さいほうではタイムスケールがある一定値、つまりは $v\sim\sigma$ で決まる値あたりになる。

この 1 次の項は、前に述べたように dynamical friction を表している。これが問題になる場面は、例えば恒星系が質量の違う 2 つの成分から出来ているような場合である。力学平衡状態で、分布関数に質量依存がないようなものを考えると、これは熱平衡から遠く離れている。従って、上の式で決まるタイムスケールで重いものがエネルギーを失い、軽いものがエネルギーを得る。これは、エネルギー等分配に向かう普通の熱力学的な進化である。

が、自己重力系ではこのエネルギー交換の結果熱平衡に近付くとは限らない。つまり、重いものがエネルギーを失い、軽いものがエネルギーを得るということは、それぞれの分布関数が変わり、空間分布も変わるということである。具体的には、重いものは中心に集中した分布になり、軽いものは外側に押し出される。その結果それぞれの成分の速度分散がどうなるかの細かいところは初期条件に依存するが、普通は物が中心に集まれば重力も強くなり、力学平衡では結局速度分散も大きくなる。このために、熱平衡からはかえって遠くなってしまう。

さて、次に、2 次の項を見てみる。速度に平行な成分も垂直な成分も、 v が大きい極限では 0 にいく。特に、 垂直な成分は v に反比例する。これに対し、速度が 0 の極限では、どちらも一定値に収束する。これはテスト粒子が停止している極限でも、回りの粒子によって揺さぶられるということを表しているわけである。

3.6 2 体緩和のタイムスケール

前節では、実際にバックグラウンドの粒子も動いている場合について、 2 体緩和によって粒子の速度、エネルギーがどう変化するかの期待値 (のモーメント) を求めた。ここでは、前回の結果を使って、まずいくつかの重要な場合について 2 体緩和がどのように働くかを議論する。

「タイムスケール」というのは、普通ある量 x の時間変化が

$$\frac{dx}{dt} = (-)\frac{x}{T} \tag{53}$$

なる形で書ける時の T のことである。もちろん、一般には T は x を含むいろんなものの関数である。

そういった意味で考えやすいのは、温度 (平均運動エネルギ) が違う 2 つの空間一様な分布が重なりあっている時に、どのようにして 2 つが近付いていくかというのものである。これは、もちろん時間が立てば熱平衡に近付くわけである。以下、実際に計算してみる。

3.7 等分配のタイムスケール :理論

今、フィールドに質量 m_fの粒子が一様に分布しており、テスト粒子として質量 m_tのものがこれもまた一様に分布しているとする。さらに、どちらも速度分布はマックスウェルで与えられるとする。ここでは等分配を考えるので、それぞれの粒子 1 個当たりのエネルギーを E_f, E_t と書く。今、テスト粒子のエネルギー変化の平均を考えると、

$$\frac{d<E_t>}{dt} = 4\pi\int v_t^2 f(v_t) <\Delta E_t> dv_t \tag{54}$$

と書けることになる。これに前回求めた $<\Delta E>$ を入れて実際に積分を実行することができて、結果は

$$\frac{d<E_t>}{dt} = 2\sqrt{\frac{6}{\pi}}\frac{m_t n_f \Gamma}{m_f}\frac{<E_f> - <E_t>}{(v_t^2+v_f^2)^{3/2}} \tag{55}$$

となる。

ここで、いくつかの極限的な場合を考えておくことは有益であろう。まず、$m_t >> m_f$で $v_t \sim v_f$ という状況を考えてみる。これはつまり非常に重いものと軽いものが、同じような

空間分布、速度分布で広がっている場合である。この時は上の式で $E_t >> E_f$ なので、

$$\frac{d\log < E_t >}{dt} = -\sqrt{\frac{3}{\pi}}\frac{m_t n_f \Gamma}{m_f v^3} \tag{56}$$

となる。なお、この時、変化率はテスト粒子の速度に分母の v_t^2 を通してしか依存しないので、 $v_t \rightarrow 0$ の極限でエネルギー変化（減速）のタイムスケールは一定値にいき、それは $v_t \sim v_f$ の時の値とそれほど違わない。

次に、$m_t \sim m_f$で $v_t << v_f$という状況を考えてみる。この時は上の式を

$$\frac{d(< E_t > /m_t)}{dt} = 2\sqrt{6/\pi}\frac{n_f\Gamma}{m_f}\frac{< E_f >}{2v_f^3} = 8\sqrt{6\pi}G^2 \ln\Lambda n_f m_f < E_f > v_f^{-3} \tag{57}$$

となる。ここで

$$\Gamma = 4\pi G^2 m_f^2 \ln\Lambda \tag{58}$$

を使った。さらに、 $E_t = m_t v_t^2/2$ などを使って書き直せば

$$\frac{d(v_t^2)}{dt} = 4\sqrt{6\pi}G^2 \ln\Lambda n_f m_f^2 v_f^{-1} \tag{59}$$

を得る。つまり、速度が小さい極限では、一定の率でエネルギーをもらうわけである。言いかえれば、温度が 2 倍になるタイムスケールというものは、温度に比例して小さくなるともいえる。

さて、通常「2 体緩和のタイムスケール」という時には何を指しているかというと、この等分配のタイムスケールのことではないのが普通である。が、時と場合によっていろんなものが出てくるが、まあ同じようなものである。普通に使われるのは、

$$t_r = \frac{1}{3}\frac{v_m^2}{< v_{\text{平行}}^2 >_{v=v_m}} = \frac{v_m^3}{1.22 n\Gamma} = \frac{0.065 v_m^3}{n m^2 G^2 \log\Lambda} \tag{60}$$

とするものである。ここで v_mは r.m.s. 速度である。1/3 になにか意味があるわけではなく、こう定義したというだけである。

これは、ローカルな量で定義されていて、例えば系全体の緩和時間といったものを考えるのにはちょっと不便なこともある。というわけで、いわゆる half-mass relaxation time t_{rh} というものを導入しておく。これは、半径 r_h の中に質量の 1/2 があるとして、その中の密度は一様であるとし、さらにビリアル定理から出てくる $T \sim 0.2GM^2/r_h$といった関係を使って出すことが出来る。ここでビリアル半径ではなく half-mass radius を使ったのは慣習に従っただけで深い意味はない。この 2 つは通常 30% 程度の範囲で一致する。これは

$$t_{rh} = 0.138\frac{N r_h^{3/2}}{M^{1/2}G^{1/2}\log\Lambda} \tag{61}$$

となる。

ここで注意しないといけないことは、 t_{rh}はあくまでも球対称に近い系の half mass radius のあたりでの緩和時間であるに過ぎないということである。従って、球状星団全体の緩和

時間とか、あるいは楕円銀河、銀河団といったものには有効な概念であるが、球対称から大きくずれた銀河とか、あるいは half mass radius のずっと外側やずっと内側では全く違ったものになっていることに注意する必要がある。さらに、速度分布が非等方であるとか、回転がメインであるとかでもまた話が全く変わってくる。このような場合、ローカルな緩和時間、あるいはエネルギー変化自体の式に戻って考えないと、タイムスケールについて全く間違った推定をしてしまうことになる。

4　熱平衡分布

なんだかよくわからないが、自己重力多体系における 2 体緩和はこんなふうに起きるということで、話をすすめる。

以下、まず、熱平衡を考える。

球対称としたので、運動の積分はエネルギー E と全角運動量 J の 2 つだけということになる。さらに熱平衡を仮定すると、速度分布がマックスウェル分布でないといけないことになる。さて、ここで、上のジーンズの定理から直ちに導かれることは、系全体の分布を記述する分布関数がエネルギーだけに依存し、これがマックスウェル分布

$$f(\mathcal{E}) = \frac{\rho_1}{(2\pi\sigma^2)^{3/2}} e^{\mathcal{E}/\sigma^2} = \frac{\rho_1}{(2\pi\sigma^2)^{3/2}} \exp\left(\frac{\Psi - v^2/2}{\sigma^2}\right) \tag{62}$$

に従うということである。ここで $\mathcal{E}$ は以下のように定義される。

$$\Psi = -\Phi + \Phi_0, \qquad \mathcal{E} = -E + \Phi_0 = \Psi - v^2/2 \tag{63}$$

Φ_0 は定数で、普通は $\mathcal{E} > 0$ で $f > 0, \mathcal{E} \leq$ で $f = 0$ となるようにとる。

球対称の定常状態を求めるための標準的な方法は、分布関数 f を速度空間で積分して密度 ρ をポテンシャルの関数として表すことである。この時に誤差関数についての

$$\frac{2}{\sqrt{\pi}} \int_0^\infty e^{-x^2} dx = 1 \tag{64}$$

を使うと、

$$\rho = \rho_1 e^{\Psi/\sigma^2} \tag{65}$$

ポアソン方程式にこれを入れると

$$\frac{1}{r^2}\frac{d}{dr}\left(r^2 \frac{d\Psi}{dr}\right) = -4\pi G\rho \tag{66}$$

従って、

$$\frac{1}{r^2}\frac{d}{dr}\left(r^2 \frac{d\log\rho}{dr}\right) = -4\pi G\sigma^2\rho \tag{67}$$

後はこれを数値的に解くわけだが、まず、一つ特別な解があるということを指摘しておく

$$\rho = \frac{\sigma^2}{2\pi G r^2} \tag{68}$$

は、上の方程式を満たし、解の一つとなっている。これを singular isothermal sphere と呼ぶ。すぐにわかるようにある半径rの内側の質量M_rを考えるとこれがrに比例するので、この解は有限質量にならない。

特別ではない解は、中心密度を有限にして中心から外側に向かって解いていけばいい。この時でも、$r \to \infty$ の極限では singular isothermal に近付く。従って、self-consisitent な解というのは所詮存在しなくて、初めに書いたように壁をつけないと意味がある解にはならない。

なお、熱平衡であるということは、実は自己重力的な流体（衝突項が大きいもの）と自己重力多体系で分布関数が同じであるということである。念のため、以下、自己重力流体について方程式を導いておく。静水圧平衡の式は

$$\frac{dP}{dr} = -\rho \frac{GM_r}{r^2} \tag{69}$$

である。状態方程式に等温の

$$P = \frac{k_B T}{m}\rho \tag{70}$$

を使ってPを消して、さらにM_rを微分してみれば、

$$\frac{d}{dr}\left(r^2 \frac{d\log\rho}{dr}\right) = -\frac{4\pi Gm}{k_B T}\rho r^2 \tag{71}$$

要するに、上と同じ方程式が出てくる。

もちろん、これは「熱平衡」という極めて強い条件を課したからで、非平衡状態を考えると自己重力流体では局所的に分布関数がボルツマン分布になるのに対し、自己重力多体系ではジーンズの定理から常に分布関数はグローバルなものであり、しかもそれは必ずボルツマン分布とは違っているわけである。

4.1 熱平衡分布の熱力学的安定性

若干前置きが長くなったが、壁が断熱壁であるとしてこの熱平衡解の振舞いがどんなものかもう少し考えてみる。上の解き方では、中心での密度、温度を与えて外に向かって解いていって、適当なところで打ち切った（そこに壁があるとした）わけだが、何が起きるかという観点からは壁の半径R、中の物質の総質量Mを固定し、全系のエネルギーEを変化させて対応する解がどうなるか調べるというのがわかりやすいであろう。で、後ですぐに明らかになる理由から、Eではなく中心での密度ρ_0と壁のところでの密度ρ_wの比$D = \rho_0/\rho_w$をパラメータにとることにする。重力がある分中心密度が高いわけで、Dの値は重力の効き方の程度に対応している。R, Mを固定した時には、$D = 1$の極限は$E \to +\infty$に対応する。重力のポテンシャルエネルギーはGM^2/Rの程度の大きさであるからである。

さて、ではDをどんどん大きくしていくとエネルギーはどうなるかというわけだが、結果だけを書くと、エネルギーはDのある値（数値的には 709 くらい）で極小になる。そのあとは振動的に$D = \infty$での値$-GM^2/4R$に近付く。

ここでチャンドラセカールによる linear series analysis を適用するなら、この $D = 709$ から先は熱力学的に不安定ということになる。[LW68] しかし、これでは不安定といってもどんなものかというのは良くわからない。不安定モードがどんなものかというのを初めて具体的に示したのは蜂巣と杉本 [HS78] である。といっても、実は彼らが調べたのは自己重力多体系ではなく流体の場合である。というのは、流体だと上に述べたように局所熱平衡で議論して良くなって、熱力学的な量を使って安定性を議論することが容易になるからである。

等温状態なのでエントロピーは通常ならば極大値である。これは、任意のエントロピーの再分配に対して、$\Delta S = 0, \Delta S^2 < 0$ となっているということを意味する。

熱をちょっとどこかからとって別のところに与えると、それによる温度変化を考えなければ（一次の変分）エントロピーは変わらない。また、温度変化を考えると（二次の変分）、普通は熱をもらった方は温度が上がっているのでもらうエントロピーは少なく、出したほうは逆に温度が下がるので出ていくエントロピーが多い、従って、系全体としては普通は摂動を与えるとエントロピーが減る、すなわち、平衡状態はエントロピー極大に相当している。

以上から、もし、熱を取り去った時に温度が上がるようなことがあればエントロピー極大ではないかもしれないということが想像できよう。もちろん、常識的な熱力学の対象ではそんなことはあり得ないわけだが、自己重力系ではそうではない。

蜂巣と杉本の方法では、摂動に対してΔS^2を求め、その符号から安定か不安定かを決めている。この方法では、しかし、熱力学的に安定かどうかをきめることは出来るが、不安定性がどのように発展するかを調べることはできない。というのは、そのためには熱伝導の式もカップルさせて線形応答を求めないといけないのに、そのような解析は行なっていないからである。というわけで、しばらく前にそういう解析をやってみたことがある [MH91] ので、以下はその結果を使う。

熱伝導の式は

$$K\frac{\partial T}{\partial r} = -\frac{L}{4\pi r^2} \tag{72}$$

と書ける。ここで、$L(r)$ は半径 r のところでの熱流束であり、K は熱伝導の係数である。K は温度、密度の関数だが、ここでは等温状態の線形解析なので温度依存性は無視してよい。従って密度だけの関数として

$$K = \rho^{\alpha} \tag{73}$$

という形を仮定する。放射伝達であれば $\alpha = -1$ である。

自己重力質点系の場合は、緩和時間の式 (60) からわかるように密度が高いほうが熱平衡に達するタイムスケールが短い。このことを熱伝導係数でむりやりに表現すると、$\alpha = 1$ となる。

エントロピーについての式は

$$\frac{\partial L}{\partial r} = -4\pi r^2 \rho T \frac{\partial s}{\partial t}|_M. \tag{74}$$

で与えられ、境界条件は

$$L = 0 \quad \text{for} \quad M = 0 \quad \text{and} \quad M = 1. \tag{75}$$

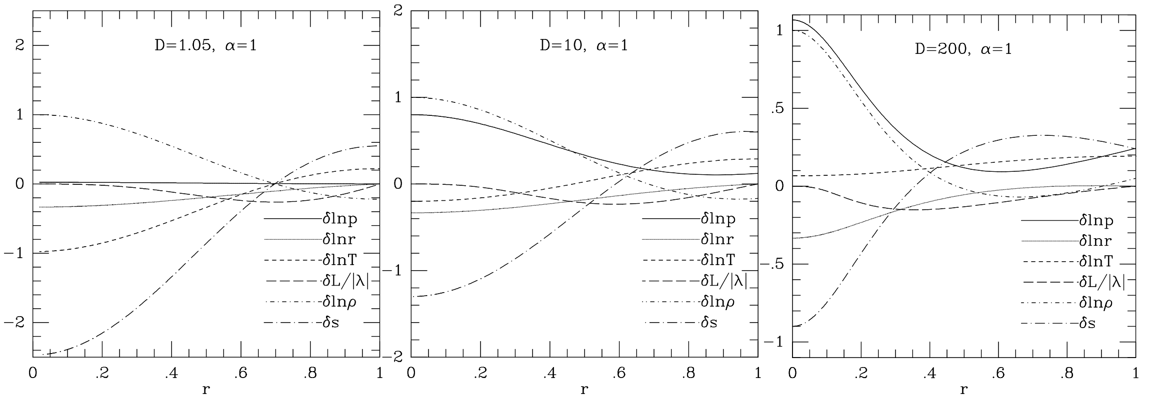

図 2: 安定領域の応答

ということになる。以下、式の細かいところを見たい人は原論文に当たってもらうとして、結果だけ書く。

図 2 に示すのは第一固有値（ここではすべての固有値が負なので、最も 0 に近いもの）に対応する固有関数である。

ここで、D は中心の密度と壁のすぐ内側での密度の比である。$D = 1$ というのは、温度が無限に高くて重力エネルギーが相対的に小さい極限である。これに対し、$D = \infty$ は singular isothermal に対応する。

$D = 1.05$ は、要するに重力が無視できる場合である。$D = 1$ の場合は応答はベッセル関数で書けるわけで、もちろんそれに近い解が求まっている。注意して欲しいことは、圧力 p の変化がないこと、エントロピー s と温度 T がちゃんと比例関係にあることである。重力が無視できるので普通の振舞いをしているわけである。つまり、密度、温度の変化が圧力変化がなくなるように働く。これは、静水圧平衡を保つためである。

なお、ここでは、中心から熱を奪って外に与えるようなものを考えているが、その逆も固有関数であることに注意してほしい。これは、線形化した方程式の解だからである。

さて、少し中心密度を上げると、摂動に対する圧力の応答が変わって、中心で圧力が上がるようになる。これは、熱を奪われることに応答して縮むと、重力も強くなるので、つじつまをあわせるにはもうすこし縮んで圧力を上げる必要が起きるからである。このために、温度の応答は与えたエントロピーからずれてくる。もっとどんどん温度をさげて、D を大きくすると、ついには、熱を奪ったにもかかわらず、温度が中心でも上昇するようになる。

もちろん、この解は負の固有値に対応するものであり、安定である。それは、温度勾配としては依然として中心に向かって下がっていて、ちゃんとエントロピー変化を打ち消す向きに熱が流れるからである。

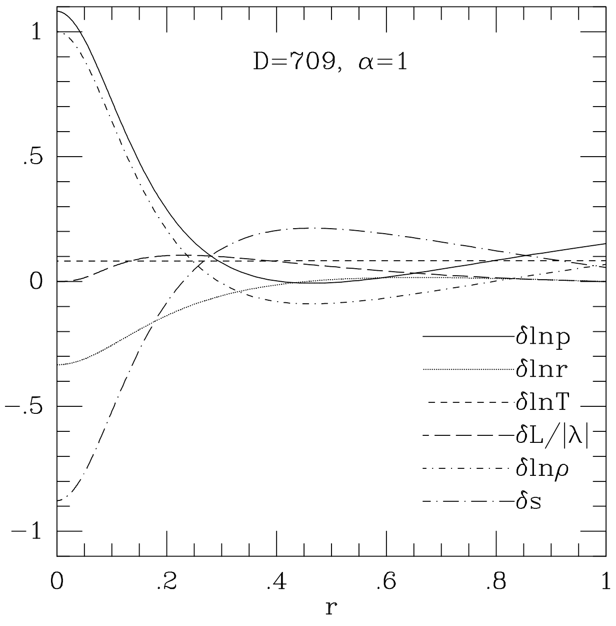

図 3: 中立応答

4.2 中立安定

さて、もっと D を大きくすると、ついには固有値が 0、すなわち与えられた摂動が減衰しなくなる。この状況を図 3 に示す。与えられた摂動が減衰しないということは、温度勾配ができないということである。実際、応答は δT が定数になっている。

4.3 重力熱力学的不安定

さらにもっと温度を下げ、 D を大きくすると、ついには固有値が正になる。図 4 にいくつかの例を示す。

どの場合でも中心でエントロピーが減っているのに温度が大きく上がり、それが外側の温度上昇を追い越している。その結果中心から外に向かう熱流ができるのである。

なお、ちょっと注意して欲しいのは、α の値によって応答が大きく違うことである。D が同じ時、$\alpha = 1$ の方が$\alpha = -1$ に比べて中心に集まったような応答になっている。これは、熱伝導が密度の高いところで速いためと考えて良い。

4.4 有限振幅での進化

さて、このあとどうなるかということを調べるためには、数値計算をする必要がある。Hachisu *et al.* [HNNS78] は、自己重力流体についてそのような数値計算を行なった。

結果の詳細は省くが、重要なことは、中心から熱をとったときに「自己相似解」が現れる

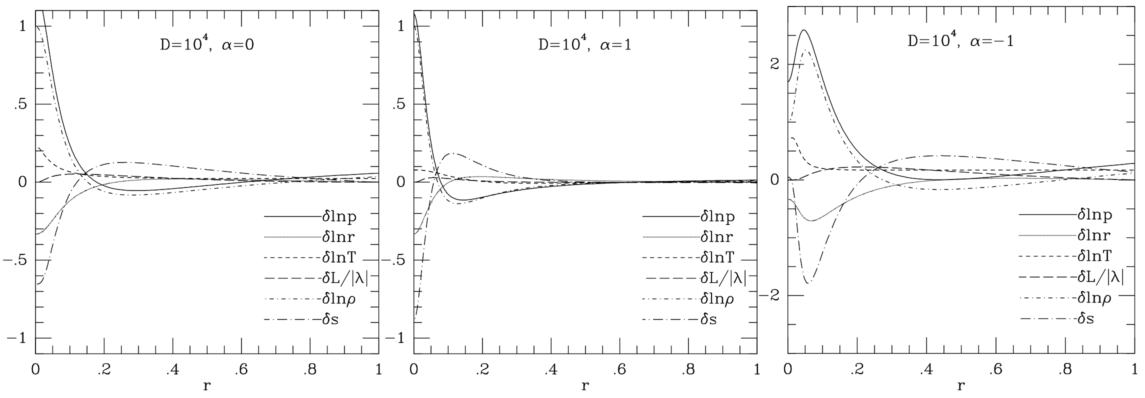

図 4: 不安定応答

場合があるということである。

中心に熱を与えると、中心は温度を下げつつ膨張する。このときは、結局最終的には安定平衡にいってしまうことになる。しかし、中心から熱をとったときにはどこかいき先があるわけではない。

この後の進化は、熱伝導のタイムスケールによる。密度が上がるとタイムスケールが長くなるような場合には、大雑把にいってかなり大きなものが全体として収縮していく。

これに対し、恒星系に対応する場合では、密度が上がるとタイムスケールが短くなる。この時は、密度の高い「コア」が出来、それがどんどん収縮を続けるということになる。これに関する詳細な解析は Lynden-Bell & Eggleton [LE80] に与えられているので、以下考え方だけを示す。

ここで自己相似解というのは、ある物理量 y が

$$y(r,t) = y_0(t)y_*[r/r_0(t)]. \tag{76}$$

と書けるようなものである。さらに、r_0 と y_0 が時間のベキで書ける（これは数値計算の結果がそうなっている）とすれば、

$$r_0 = (t_0 - t)^{\beta}, \tag{77}$$

とか

$$y_0 = (t_0 - t)^{\gamma}, \tag{78}$$

と書け、結局

$$y_0 = r_0^{\gamma/\beta}. \tag{79}$$

という関係が出てくる。

自己相似解ということで、いろんな無次元量は一定と考えられる。特に、今コアというものを考えて、その半径を r_c とすれば

$$\sigma^2 \propto \frac{GM_c}{r_c} \sim \rho_0 r_0^2. \tag{80}$$

ここで ρ_0 を

$$\rho_0 = r_0^{\alpha}, \tag{81}$$

と書けば

$$r_0 = (t_0 - t)^{2/(6+\alpha)}. \tag{82}$$

となる。

実際に $r_0(t)$ とかを求めるには、やはり固有値問題をとくことになる。Lynden-Bell & Eggleton は実際にといて、

$$\rho_0 = r_0^{-2.21}. \tag{83}$$

という答を得た。

4.5　ガスと N 体の違い

実は、このあたりの進化、すなわち重力熱力学的不安定や自己相似解については、ガス近似、フォッカー・プランク近似を使って分布関数の進化を数値計算する方法、N 体計算の間の一致は素晴らしくよい。これが何故かというのは本当にわかってるのかといわれると困るようなものだが。

ガスではうまく表現出来なくなるのは、質量分布がある場合、非等方性が発達する場合等である。

4.6　自己相似解の後の進化

さて、自己相似解は、ある時刻 t_0 で密度が無限大になる。これを collapse と呼んでいる。実際にそんなことが起きるのか、また、そのあとはどうなるのかというのは現実的には重要な問題である。というのは、多くの球状星団、あるいは小さな銀河では、タイムスケールを見積もるとすでに collapse しているはずだからである。

その後どうなるかについては、いろんな可能性が考えられた。特に、これによってブラックホールを作るというアイディアはそれなりに真剣に検討されたし、まだ完全に見捨てられたわけではない。特にできたばかりの若い星団を考えると、現在の球状星団には存在しない大質量の星がある。緩和時間は質量に反比例するので、こういった系では進化が速いし、大質量星は dynamical friction によって中心に沈むのでさらに進化が速くなる。この場合には暴走的に合体が進んでブラックホールができる可能性があることが最近の研究では示唆されている。[PMMH99]

しかし、現在のところ、典型的な球状星団や銀河では、ブラックホールが出来るというのはありそうにないとされている。コアが十分に小さくなると、エネルギー供給源が出来るからである。

ここでのエネルギー供給の元は連星である。仮に星団があらかじめ連星をもっていなかったとしても、コアが十分に小さくなると、そのなかで 3 体相互作用で連星ができるように

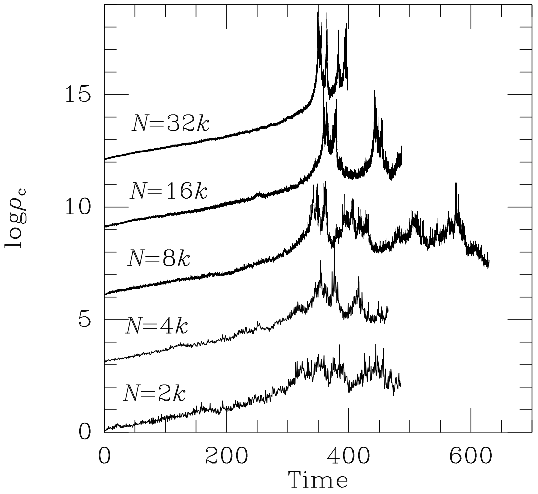

図 5: 重力熱力学的振動

なる。これは基本的には星のなかで温度、密度が上がると核融合が始まるというのと変わるところはない。ただし、量子力学的な効果やポテンシャル障壁はないので、連星の出来やすさは密度と温度 (平均速度) の関係だけで決まる。

連星によるエネルギー供給が始まると、コアの収縮は止まる。熱源として連星を考えた計算を初めて行なったのはエノン [Hen75][1] であり、1982 年ころまでにいくつかそのような計算が行なわれた。それらでは、コアからの熱伝導による熱の流出と連星からのエネルギー入力がバランスし、系全体がホモロガスな膨張をするという結果が得られていた。

しかし、1983 年になって、杉本とベトウィーザー [SB83] は、実はこのホモロガスな膨張解も熱力学的に不安定であるという発見をした。そのあと数年に渡る論争があったが、1985 年には他のグループによるガスモデル計算、1986 年には FP 計算でも振動が確認された。実際に粒子系でそんなものが起きるかどうかにはさらに議論があったが、1995 年になって N 体数値計算でも確かに振動が起きるということが見い出された [Mak97]。図 5 に結果を示す。縦軸は中心密度である。粒子数が違う計算結果は適当に縦にずらしてある。横方向は、緩和時間がほぼ等しくなるようにスケールした時間である。

粒子数が小さいものではよくわからないが、大粒子数の計算では大振幅の非線形振動がおきているのがはっきりわかる。

4.7 現実的な系

前節で見たように、極めて単純な自己重力系、すなわち、同じ重さの質点がおおむね球状に集まっている系については、その熱力学的な進化の過程がどんなものかというのはほぼ明らかになったと考えてよい。しかし、実際の天体現象を考えると、

---

[1] エノン・ハイレスのカオスやエノン・マップで有名なエノンと同一人物である

- 多くの系は年齢が熱力学的進化のタイムスケールよりはるかに若い
- 回転や速度の非等方性の効果は無視できない
- 構成要素である星の質量は等しくない
- 星は質点ではなく、大きさをもつ
- 星間ガスとかそういうものもないわけではない
- 相対論的効果とか、、、

と、いろんなことがあるわけである。これらの効果を見ていくためには、前節で得られたような基本的な描像を手がかりとしつつ、現実的な効果を採り入れた数値計算によって調べていくということになる。

というわけで、ここから数値計算の方法についての話にしたい。

5 数値計算の方法 — 前置き

数値シミュレーションといっても、方法は1通りではない。前節で見たように怪しげな流体近似を使って定性的な議論をするとか、粒子数無限大でのフォッカー・プランク近似から得られる分布関数についての偏微分方程式を数値積分する方法もある。

これらに対して、多体系の運動方程式を積分する多体シミュレーションの方法は、あくまでもいくつかある方法のうちの一つでしかない。

とはいえ、他の方法に比べた利点があることも確かではある。極端な話、銀河系の星の数と同じだけの粒子数を使えば、少なくとも銀河系の自己重力多体系という側面については（ガスがどう振舞うとか、磁場の影響がどうかとかいうことを別にすれば）、「正しい」計算結果を得ることができるはずである。これに対し、他の方法ではそこで明示的に、あるいは暗黙のうちに導入される仮定すべてが正当化出来るわけではなく、「この計算結果は正しいか、そもそも正しいというのはどういう意味か」というナイーブな疑問に答えるのはそれほど容易なことではない。

とはいえ、現在我々が利用可能な計算機の速度と我々が知っている計算法では、銀河系の星（とそのほかのダークマターとか）を粒子毎に表現するというのは不可能である。[2] もちろん、現実の系と同じだけの粒子数を使わないと意味がある結果がでないというわけではない。粒子数が少ない計算でも、定性的には有意味な結果が得られるし、タイムスケールのスケーリング等で理論的には小規模な計算の結果を現実の系や粒子数無限大の極限にもっていけるはずである。しかし、ここでは結局「この計算結果は正しいか、そもそも正しいというのはどういう意味か」という疑問が戻ってきてしまう。

まあ、そんなある意味哲学的ないい方をするまでもなく、現在の計算機で出来る最大規模のシミュレーションで結果が正しいかどうかぎりぎりのところが現在の研究の最先端であるわけで、より高速・高精度・大粒子数の計算を可能にすることで新しい領域が開けるわけである。というわけで、以下ではそれにはどういう方法があるか、また今後の研究の方

---

[2] 計算量の他に、ダークマターはそもそもどういうものでどれくらい質量があるかもわからないという原理的な問題もあることはあるが、、、

向はどういうものかという話をする。

計算法を改良するのとより速い計算機を使うというのが基本的な方向である。このバリエーションとして、速い計算機を使うためのに計算法を改良するというのもあるし、また自分で速い計算機をなんとかするという方向も考えられる。

5.1 何が問題か？

原理的には、多体シミュレーションは非常に単純なものである。つまり、以下の運動方程式

$$\frac{d^2\boldsymbol{x}_i}{dt^2} = -\sum_{j\neq i} Gm_j \frac{\boldsymbol{x}_j - \boldsymbol{x}_i}{|\boldsymbol{x}_j - \boldsymbol{x}_i|^3}, \tag{84}$$

を数値積分するだけである。

これをプログラムにするだけなら、時間積分に何を使うかによるが大したことはない。右辺を計算するプログラムは 2 重ループで 10 行くらいである。というだけで話が済めばいいけれど、もちろん世の中はそんなに簡単ではない。

大きな問題は 2 つある。一つは、計算精度の問題である。これもさらにいろいろあるが、主要なものは次の 2 つである。一つは前に述べた自己重力による構造形成である。このために、時間がたつと中心密度が上がる。時間積分という観点からすると、時間刻みを決めるのは粒子の軌道運動のタイムスケールであり、粒子の軌道運動のタイムスケールは基本的には自由落下時間

$$t_{\rm ff} = 1/\sqrt{G\rho} \tag{85}$$

で与えられる。これは以下のように出てくる。密度 ρ、半径 R の球を考えると、その表面での重力ポテンシャルは $-G\rho R^2$ の程度であり、粒子の典型的な速度はビリアル定理から単位質量あたりの運動エネルギーとポテンシャルが同じ程度ということから、$\sqrt{G\rho R^2}$になる。これで系の大きさ R を割れば上の $t_{\rm ff}$ が出てくることになる。まあ、この出し方はだいぶ適当だが、真面目に一様密度球のなかでの軌道を計算すれば、これは単なる調和振動子なので周期はすぐに出て、係数を別にすれば同じ結果になる。もちろん、この $t_{\rm ff}$ は前に出てきた力学的な時間スケール t_d と係数を別にして同じものである。

つまり、問題が何であったかというと、タイムスケールが短い領域が出来てくると、すくなくともその領域の粒子は時間刻みを必要に応じて短くしなければいけない。が、ここで、なにも考えないで全粒子の時間刻みを短くしては計算時間が増え過ぎる。どの程度計算量が増えるかはどの程度空間構造が発達するかによるわけであり、これを言い換えると利用可能な計算機資源によってどこまでの空間構造の発達を計算できるかが決まるということになる。

もう一つの問題は、それほど空間構造が発達していない系でも、重力が引力であるために確率的には 2 つの粒子がひじょうに近付くような近接散乱が起こるということである。前節で扱ったように、粒子数が有限の系では各粒子は他の粒子との近接重力散乱を通してエネルギーを交換し、系は熱力学的に進化していく。ここで、理論的な問題としては粒子同士の衝突のインパクトパラメータ b の上限はどこかということが問題であった。インパク

トパラメータが 0 でも、質点とすれば直線解があるわけであり、普通の衝突と同じように速度が反転するだけであるので、2 体緩和への寄与は有限なので、こちらでは積分は発散しない。

だが、多体系の数値積分においてそのような軌道をどう扱うかというのは大きな問題である。ナイーブな方法は、もちろん一番短いタイムステップを必要とする粒子に合わせてタイムステップを決めるということである。しかし、これではもちろん大きな無駄が発生することになる。これは計算精度の問題というよりは計算量の問題である。

もちろん、数値計算においては常に計算精度と計算量の間にはトレードオフがあるわけであり、計算精度の問題と計算量の問題というふうに明確に分離出来るわけではない。

といいながら、計算量の問題という方に話を移す。これは、運動方程式の右辺の計算量が $O(N^2)$ であり、粒子数が大きくなると急速に計算量が増えるということである。

もうちょっと状況を整理すると、これらは問題によってというよりは解決法によって分類されているということができる。つまり、時間領域で効率化を図る方法と、空間領域で効率化を図る方法があるということに今までの説明は基本的には対応している。

というわけで、以下、時間領域での計算法と空間領域のそれを概観しよう。

6　計算法 — 時間領域

6.1　独立時間刻み

重力多体問題は、原理的には単に大規模な常微分方程式の初期値問題である。というわけで、普通に考えると、数値計算ライブラリパッケージに入っているルンゲ・クッタ法や線形多段階法のサブルーチンをそのまま使えばいいような気もするが、これで話が済むことはほとんどない。これは、既に述べた 2 つの問題に対応しないといけないためである。

実際に、重力多体問題を扱う上で鍵となる手法は 2 つある。一つは独立時間刻みであり、もう一つは安定な連星系をサブシステムと認識し、それに対しては特別な扱いをする方法である。

独立時間刻みというのは、基本的には粒子毎にばらばらの時刻 t_i と時間ステップ Δt_i を与える方法である。時間積分は、以下のような順序で進むことになる。

1. $t_i + \Delta t_i$ が最小の粒子を選ぶ。
2. その粒子の軌道を新しい時刻まで積分する。
3. その粒子の新しい時間刻みを決める。
4. ステップ 1 に戻る。

と、こう書くと、もちろん謎が発生する。軌道を積分するといっても、それにはなんらかの方法で他の粒子からの重力を計算する必要がある。つまり、その粒子の新しい時刻での他の粒子の位置が必要になる。しかも、例えばルンゲクッタのような方法では、現在の時刻、新しい時刻の他に、その中間のいくつかの時刻でも他の粒子の位置が必要になる。

例えば、時間積分が前進オイラー公式のような 1 次精度のものであれば、これをどのように解決すればいいかというのは自明である。つまり、別に何もしなくていい。1 次精度の範囲内では、ある粒子の新しい時刻での加速度を計算するのに、他の粒子についてはそれらのそれぞれの時刻での位置をそのまま使ってかまわないからである。

ここで、とりあえず満たすべきは積分公式の次数、すなわちタイムステップの何乗で誤差が小さくなるかということだけであるとすると、とりあえず、以下のような方針が考えられる。

- 各粒子の時間積分公式としては、可変ステップの線形多段階法を PEC モードで使う。(いわゆる予測子・修正子法)
- ある粒子の新しい時刻での加速度を計算するには、他の粒子のその時刻での位置を予測子を使って予測する。

といってもコトバを知ってる人にしかわからないような気がするので、以下説明。

6.2 線形多段階法

まず線形多段階法とは、常微分方程式の初期値問題の解法の一つである。ルンゲ・クッタとの違いは、何ステップか前での導関数の値や数値解そのものをとっておいて、それらの線形結合として新しいステップでの解を求める方法である。

原理的には非常にいろいろな方法があり得るが、現在実際に使われているのはほとんどアダムス型公式といわれるものでる。この方法では、新しい時刻 t_{i+1} (ここでは添字は粒子の番号ではなく時刻を表すことにする) での解 x_{i+1} を、前の数ステップでの導関数から作った補間多項式 $P(t)$ を t_i から t_{i+1} まで積分して求める。つまり

$$x_{i+1} = x_i + \int_{t_i}^{t_{i+1}} P(t)dt \tag{86}$$

補間多項式の作り方にも無限にいろいろありえるが、実際に使われているのは $t_{i-p}, t_{i-p+1}, ...t_i$ と $p+1$ 点とって p 次多項式を作る方法と、$t_{i-p+1}, t_{i-p+2}, ...t_{i+1}$ の $p+1$ 点をとって同じく p 次多項式を作る方法である。前者の方法では、x_{i+1} を作るのにそこでの導関数の値は必要ないので、これは陽解法ということになる。後者では x_{i+1} を求めるのにそこでの導関数が必要になるので、これは陰解法である。陽公式の利点はプログラムが簡単でステップ当たりの計算量も少ないことであり、陰公式の利点は安定性、誤差の両方がステップサイズが同じなら陽公式に比べて圧倒的によいということである。陽公式はアダムス・バシュフォース公式、陰公式はアダムス・モールトン公式という名前がついている。

余談になるが、アダムスは 19 世紀末のイギリスの数学者 (物理学者・天文学者) であり、彗星の軌道の数値計算のためにこの方法を開発したそうである。

実際には、陽公式と陰公式を組み合わせて使う方法が実装も容易で便利である。つまり、最初に陽公式を使って x_{i+1} を推定し、それを使って導関数を計算し、計算した導関数を使って今度は陰公式を適用して x_{i+1} を計算しなおす。原理的には、さらに導関数を計算し、それを使って x_{i+1} を計算しなおすのを、x_{i+1} が収束するまで繰り返さないと陰的解法にはな

らない。が、通常の場合には、反復を全くやらないでおしまいとか、あるいは一度だけやっておしまいにする場合が多い。

これは単に用語の説明だが、反復を全くやらない方法が PEC モードというものである。これは、陽公式の適用のことを predictor step（P, 予測）、導関数の計算を evaluate step (E, 評価)、陰公式の適用を corrector step (C, 修正) と呼ぶことにして、それらを一度ずつこの順番でやるという意味である。これに対して、1 度反復を繰り返す方法は $\mathrm{P(EC)}^2$ モードと書かれる。原理的には PE とか PECE $\mathrm{PE(CE)}^2$とかいった、E で終る方法もあり得る。

普通の数値計算の教科書等を見ると、$\mathrm{P(EC)}^2$ や $\mathrm{PE(CE)}^2$のような公式が勧められていることが多いが、これは（後に述べる対称型公式のような）特別な場合を除けば重力多体問題ではあまり得にはならない。これらのほうが、真の収束値に近い分ある程度誤差が減ることは確かである。しかし、計算量のほとんど全てが加速度の計算にかかるので、タイムステップが同じなら計算量がほぼ 2 倍になる。それよりも PEC モードでタイムステップを半分にしたほうが得になることが多い。

なお、次数についてだが、これはもちろん粒子数、要求する精度等、さまざまな条件によって変わってくる。例えば太陽系の惑星の長時間積分というような話になると、12 次とか 14 次といった超高次の公式が使われる。これに対し、球状星団や銀河中心核といった問題では 4 次程度の比較的次数の低い公式が使われることが多い。理論的にも実験的に要求する計算精度に依存して最適な次数が存在するわけだが、現状では最適な次数というよりは実装の単純さ等によって次数を決めている面もある。

なお、この方法でプログラムを実際に書くのは結構面倒である。この後で、もうちょっとプログラムが書けるような方法について議論するが、基本的な考え方というか、キーワードは示しておこう。タイムステップと次数が固定の場合は話は簡単である。アダムス法の場合は、結局 $\Delta x = x_{i+1} - x_i$ が導関数の値の線形結合で書けるわけなので、その係数をあらかじめ求めるか、適当な数値解析の教科書から写してくればいい。

6.3 可変時間刻みの実現

可変刻みにすると話が急にややこしくなる。線形結合で書けるといっても、その係数が毎回変わるからである。実際のプログラムでは、まず不等間隔補間多項式を構成し、それを代数的に積分（もちろん多項式なので各項を次数+1 で割るだけだが）し、求まった積分された多項式にタイムステップをいれてΔx を求めるという手順になる。

補間多項式を構成する標準的な方法は 2 つあって、一つは差分商 (divided difference) を使ってニュートン補間多項式を作る方法である。これは、もちろん求まる多項式は一意に決まるのでラグランジュ補間と同じだが、数値的に安定であるとか、また補間範囲を一点ずつずらしていくときに前の計算結果の使い回しによって計算量を減らせるとかいった利点があり、名前からも想像できるように古くから使われている補間多項式の構成法である。

もう一つは、通常 multivalue method または提案者の名前をとって Nordsiek method といわれている方法である。この方法では、差分商の代わりに補間多項式の係数、つまり高階の導関数の値そのものを記憶しておく。これの良いところは、見かけ上 1 段階法になって

いるので気分的にタイムステップを変えるのが簡単であるということである。なお、実際には、見かけ上 1 段階法であるが、補間多項式をずらすときには、考え方としては等間隔補間の元の参照点での値を作り直し、もっとも古い点を落してそのかわりに新しい点を付け加えて補間多項式を作りなおすというような手間が必要になる。タイムステップが頻繁に変わる場合には、multivalue method では補間多項式の性質が多少悪くなるという問題が少なくとも理論的には存在する。

以下、もう少し積分公式の話を続ける。

6.4 エルミート公式

まず、細かい話であるが、ハミルトン系のような 2 階の方程式の場合には、アダムス型公式をそのまま使うのはあまり賢い方法とはいえない。通常、高階の微分方程式の数値積分をするには、最高次の一つ下までの導関数を変数にとって連立の一階微分方程式系に置き換える。が、アダムス法の場合、加速度に対する補間多項式を 2 回積分すれば位置の変化が求まるわけであり、速度についての補間多項式を構成したり、速度の昔の情報をとっておく必要はない。このため、必要なメモリ、計算量の両方が減る。

もう一つ、線形多段階法、あるいは一般に常微分方程式の数値解法のほとんど全てについていえることだが、良く研究されている解法では基本的には微分方程式を与えられたものと考え、その右辺値、つまり導関数だけを使って積分公式を構成する。

多くの教科書で、高次の導関数を使って直接テイラー展開を計算して近似解を構成する方法が「テイラー法」として紹介はされるが、通常高次の導関数を計算するのに必要なコストは次数の指数関数で上昇するのであまり現実的な方法ではないとしてそれ以上詳しい議論はなされない。

しかし、一般に非常に高い次数まで計算すれば指数関数的に高価になるのは確かだが、例えば次の導関数くらいならそれほど大変ではないことが多い。例えば、2 体間の重力

$$\boldsymbol{f}_{ij} = -Gm_j \frac{\boldsymbol{r}_i - \boldsymbol{r}_j}{\boldsymbol{r}_{ij}^3} \tag{87}$$

なら、その一階導関数は

$$\boldsymbol{j}_{ij} = -Gm_j \left[\frac{\mathbf{v}_{ij}}{r_{ij}^3} - \frac{3(\mathbf{v}_{ij} \cdot \mathbf{r}_{ij})\mathbf{r}_{ij}}{r_{ij}5} \right], \tag{88}$$

である。ここで

$$\begin{aligned} \boldsymbol{r}_{ij} &= \boldsymbol{x}_j - \boldsymbol{x}_i, \qquad (89) \\ \boldsymbol{v}_{ij} &= \boldsymbol{v}_j - \boldsymbol{v}_i. \qquad (90) \end{aligned}$$

であり、$\boldsymbol{x}_i$、$\boldsymbol{v}_i$はそれぞれ粒子 i の位置、速度、G は重力定数、m_jは粒子 j の質量である。式はちょっと複雑だが、追加計算には除算や平方根演算は必要ないので、計算時間の増加はたいしたことはない（計算機によるが、2 倍以上になることは稀である）。これ以上高階の導関数を計算するのは、さすがに現実的ではない。

このように微分方程式の右辺の時間導関数が比較的安価に計算できるので、アダムス法を拡張して、補間多項式を導関数も使って構成することが考えられる。この型の補間自体はエルミート補間と呼ばれる。エルミート補間を使う積分公式は、人によってエルミート型、あるいはオブレヒホフ型の公式と呼ばれる。この公式自体は古くから知られていて、数値解析の教科書を見ると載っているものもある。が、まあ、一応、重力多体問題にこの型の公式を導入したのは著者であるらしい。[Mak91b]

この方法の利点は、段数に対して達成出来る多項式の次数が2倍になることである。特に、実質的には1段法である2段の陰的公式で4次精度が実現出来るため、多くの応用ではこれで十分な精度が達成出来る。

ただし、この場合予測子は2段とはいえやはり多段階法であり、実装はけっこう面倒である。ここで、さらにもう少し手を抜くために、予測子の最高次の次数を位置の時間に対する3次までで止めるという方法が考えられる。誤差の係数は別にして、公式の次数に関する限りは、これでもつじつまはあっている。

このやり方の大きな利点は、出発公式が不要であるということである。線形多段階法一般にいえることだが、公式自体のプログラムよりも、最初に時間積分を始めるところをどうするかのほうが面倒である。というのは、公式は加速度の過去の値を必要とするのに、積分を始める時にはもちろんまだ過去の情報は持っていないからである。

一般には、この場合にどうするかというと2通りの方法がある。もっとも一般的な解決法は、タイムステップだけでなく次数も可変な公式を使うことである。最初は次数2の公式から出発して、（必要に応じて短いタイムステップにしておき）、積分を進めて過去の値が入るのに応じて次数を上げていく。

次数が4程度と低い場合には、もっと直接的な方法もある。例えば、広く使われているプログラムでは、加速度の3次までの導関数を直接計算し、それから過去の差分商を構成する方法が使われている。

というわけで、一般の線形多段階法で2次以上になると出発公式をどうするかを考えないといけないが、エルミート公式では4次でも出発公式が不要になる。これは手間を省くという点からは馬鹿にならない。

以下、この場合に積分公式がどうなるかをまとめておく。

予測子は単に以下のようなテイラー展開で与えられる。

$$\boldsymbol{x}_{\rm p} = \frac{\Delta t^3}{6}\dot{\boldsymbol{a}}_0 + \frac{\Delta t^2}{2}\boldsymbol{a}_0 + \Delta t\boldsymbol{v}_0 + \boldsymbol{x}_0 \qquad (91)$$

$$\boldsymbol{v}_{\rm p} = \frac{\Delta t^2}{2}\dot{\boldsymbol{a}}_0 + \Delta t\boldsymbol{a}_0 + \boldsymbol{v}_0, \qquad (92)$$

ここで $\boldsymbol{x}_{\rm p}$ と $\boldsymbol{v}_{\rm p}$ はそれぞれ位置と速度の予測子であり、$\boldsymbol{x}_0, \boldsymbol{v}_0, \boldsymbol{a}_0$ および $\dot{\boldsymbol{a}}_0$ は前の時刻での位置、速度、加速度と加速度の一階時間導関数である。Δt は時間刻み $t_1 - t_0$ である。

この位置と速度の予測子を使って新しい時刻 t_1 で加速度とその導関数$\boldsymbol{a}_1$ および $\dot{\boldsymbol{a}}_1$ を計算

する。これらを使って、時刻 t_0 での加速度の 2 階と 3 階導関数は以下のように求められる。

$$
\begin{aligned}
\boldsymbol{a}^(2)_0 &= \frac{-6(\boldsymbol{a}_0 - \boldsymbol{a}_1) - \Delta t(4\dot{\boldsymbol{a}}_0 + 2\dot{\boldsymbol{a}}_1)}{\Delta t^2} \\
\boldsymbol{a}^(3)_0 &= \frac{12(\boldsymbol{a}_0 - \boldsymbol{a}_1) + 6\Delta t(\dot{\boldsymbol{a}}_0 + \dot{\boldsymbol{a}}_1)}{\Delta t^3}
\end{aligned}
\tag{93}
$$

これから、修正子を書き下して整理すると、

$$
\boldsymbol{x}_{\mathrm{c}} = \boldsymbol{x}_0 + \frac{\Delta t}{2}(\boldsymbol{v}_{\mathrm{c}} + \boldsymbol{v}_0) - \frac{\Delta t^2}{12}(\boldsymbol{a}_1 - \boldsymbol{a}_0), \tag{94}
$$

$$
\boldsymbol{v}_{\mathrm{c}} = \boldsymbol{v}_0 + \frac{\Delta t}{2}(\boldsymbol{a}_1 + \boldsymbol{a}_0) - \frac{\Delta t^2}{12}(\dot{\boldsymbol{a}}_1 - \dot{\boldsymbol{a}}_0). \tag{95}
$$

ということになる。なお、ここでは、最高次の項は速度の修正子にだけ使って、位置と速度を同じ形に書き下した。最高次の項を位置に入れても、時間積分の精度は上がらない。独立時間刻みにするなら、他の粒子の予測子も計算しないといけないが、これらも単にテイラー展開を評価するだけなのでなんということはない。

6.5 時間刻み自体の決定

あと、決めないといけないのは、タイムステップの計算式である。伝統的には、以下のなんだか意味が良くわからない形の式が使われている。

$$
\Delta t = \sqrt{\eta \frac{|\boldsymbol{a}_1||\boldsymbol{a}^(2)_1| + |\dot{\boldsymbol{a}}_1|^2}{|\dot{\boldsymbol{a}}_1||\boldsymbol{a}^(3)_1| + |\boldsymbol{a}^(2)_1|^2}} \tag{96}
$$

これは、どういう理屈があって決まるのかといわれても困るような代物だが、経験的には非常にうまく動くことが知られている。以下、「うまく」動くとはどういうことかをもうちょっと考える。

6.5.1 誤差評価と誤差の指数的成長

理屈としては、時間ステップをどのように決めるべきかというのは結構良くわからない問題である。もちろん、位置なり速度の誤差の推定は出来るわけで、その誤差をある値以下になるように調整しようというのが普通の話だが、ここで問題になるのが空間構造が発達し、タイムスケールに大きな幅が出来るということである。このために、例えば位置や速度の絶対誤差に上限をつけるというのがよい方法かどうかはあまり自明ではない。といっても、ではといって相対誤差にすればよいというものでもない。つまり、絶対誤差にすれば、特に速度が小さい粒子に対して相対誤差が非常に大きいということになる。ではといって相対誤差にすると、まず誤差がガリレイ変換に対して不変でないという問題が生じる。例えば、系が全体として運動していると、そのなかでの粒子の運動がどちら向きかで時間刻みが変わることになってこれは非常に変である。また、もちろん、速度が大きいと絶対誤

差としては大きくなるわけで、例えばエネルギー保存を重視するなら、誤差のほとんどが速度の大きいところから出るということになってなんとなく無駄なような気がする。

つまり、原理的には、多体計算において「どのような誤差を最小にするべきか」という問題があるわけだが、これは、要するに、良くわかっていないのである。もちろん、良くわかっている問題もある。例えば、太陽系の惑星の運動の長時間積分という場合には、最終的に問題になるのは例えば 10 億年積分したあとでのそれぞれの惑星の位置そのものであり、それらの誤差が許容範囲内になるようにタイムステップを決めることになる。また、逆に、非常に大きなスケールでの宇宙の構造形成といった場合にも、観測と比較するのは例えば銀河団の空間分布の 2 体相関関数であり、さらに、その構造は自由落下のタイムスケールで出来たものである、つまり、いい方を変えれば、そのスケールでの構造は一様な宇宙膨張から密度揺らぎが自己重力で成長することで出来るわけだが、これはまだ成長中であって共動座標でみてものはあまり動いていないという事情があるので、ここでも位置、速度そのものを見るというのはそれほど悪いやりかたではない。

問題は、この 2 つの中間の場合である。この領域では、シミュレーションの最終状態でのそれぞれの粒子の位置は原理的にほとんど意味を持たない。これは、各粒子の軌道そのものは、リヤプノフ指数が正で、その時間スケールが自由落下時間 $t_{\rm ff}$ (式 85) の数分の一程度であるという意味でかなり強くカオス的であるからである。これは、大雑把にいえば有効数字が $t_{\rm ff}$ あたり 1 桁程度減っていくということであり、従って $10\ t_{\rm ff}$ もたてば初期条件の精度が 64 ビット浮動小数点数の有効数字だけあったとしても、最終状態での個々の粒子の位置の精度は 10 桁落ちて 5-6 桁しか残っていない。つまり、2 つ全く同一な系で、1 つの粒子だけ初期条件の最後の 1 ビットだけ変えて計算を始めたとして、$10\ t_{\rm ff}$ 程度でそのずれが 10^{10} 倍にも増幅されるわけである。

これは、力学系自体の性質なので、どれほど数値積分を正確に行なっても同じことである。球状星団や散開星団の進化の計算では、数千から数万クロッシングタイム程度の長さのシミュレーションを行なうが、その後で各粒子の位置が意味を持つためには計算精度が数万桁いるわけである。そんなことはできるはずがないので、多体シミュレーションでは必ず個々の粒子の位置は「正しく」ない。

このように指数関数的な成長が起きるのは、難しげにいうと Γ 空間（$6N$ 次元位相空間）での軌道が双曲的であるからということになるが、もちろんこれは指数関数的に広がるというのの言い替えでしかない。具体的にミクロスコピックにみてどうやって広がるかというのを見てみると、これは結局他の粒子と近接遭遇した時に軌道間の距離が拡大される傾向があるからである。2 次元の時のイメージを図 6 に示す。

ここで、細かい人は 3 次元だと紙に垂直な方向では軌道が縮んでいるのではないかと思うかもしれない。軌道のずれの 1 次（線形）の分だけをとって計算するとそうなるが、ここでは実は 2 次の項の分が効き、そちらは拡大する方向に働くのである。

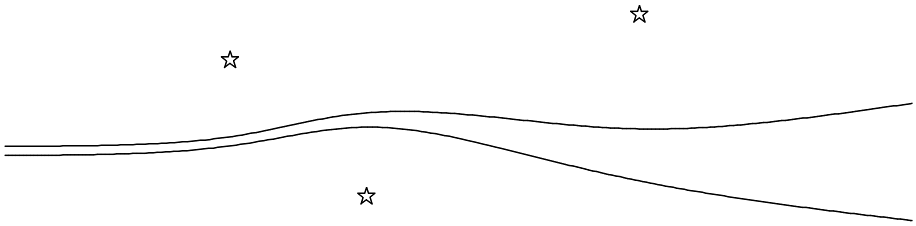

図 6: 軌道間距離の指数関数的拡大

6.5.2 指数的成長と熱力学的緩和

なお、ここで時々誤解 [GS86] があるような気がするので断っておくが、このように初期条件の差が指数関数的に成長する時間スケール（大雑把にいって最大リアプノフ指数の逆数程度）と、系が熱力学的に緩和する時間スケールとの間にはなんの関係もない。実際、前に述べたように熱力学的に緩和する時間スケールはほぼ粒子数に比例するのに対し、指数関数的成長の時間スケールは粒子数に無関係に一定なのである。さらに、指数関数的成長を担うのは、上に述べたように比較的近接した粒子同士の相互作用である。具体的には、距離が $R/\sqrt{N}$ の程度のものの寄与が最大である。この程度の距離まで粒子同士が近付くのはクロッシングタイムに 1 度程度であり、これは粒子数 N に無関係である。これに対し熱力学的緩和を担うのは R/N から R までの全ての粒子との相互作用であり、従って近距離でのポテンシャルの形をすこし鈍らせることで、熱力学的緩和の時間スケールをあまり変えないで指数関数的成長の時間スケールのほうは非常に長くすることができる。

ここでよろしくないのは、熱力学的緩和とは「系が初期条件を忘れること」であり、また、初期条件の微小なずれが指数関数的に成長することがまさにその「忘れる」ことであるとなんとなく思ってしまうことである。

少し思考実験をしてみよう。今、非常に粒子数が大きい系を考え、そのなかで、ある位相空間の微小体積を考える。熱力学的緩和によって初期条件を忘れるとは、基本的にはその中にある粒子が系分布関数全体（というか、まあ、とにかくある程度の広さ）に広がるということであろう。それらが依然としてどこかに固まっていれば、緩和したとはいい難い。では、指数関数的な成長でこの微小体積をどのように変形できるのであろうか？配位空間での大きさが、上の $R/\sqrt{N}$ よりも小さいうちは、空間の少なくとも 1 方向に向かってはほぼ指数関数的に伸びていく。これは、他の粒子との散乱が基本的にはコヒーレントであるからである。つまり、微小体積の大きさに比べて散乱のインパクトパラメータが大きく、微小体積のなかの各粒子の軌道は基本的には同じように曲がる。同じように曲がるので、その差を考えると散乱するほうの粒子に近いほうが大きく、遠いほうは小さく曲がり、差が指数関数的に成長できるのである。

逆に、どんどん延びていってある 1 方向に対して長さが $R/\sqrt{N}$ を超えると、散乱のインパクトパラメータに比べて微小体積のほうが大きいような散乱が頻繁に起きることになる。このような散乱では、上の場合のように線形に微小体積が引き延ばされるというような簡単な話にはならない。むしろ、長さが $R/\sqrt{N}$ よりも十分大きい極限では、各粒子がどのよ

うな散乱を受けるかというのは他の粒子とはほとんど無関係になる。もちろん、非常にインパクトパラメータの大きな散乱に対しては依然としてコヒーレントなわけだが、それによる指数関数的成長の時間尺度はインパクトパラメータが大きくなるにつれて急速に長くなる。こうなると、各粒子間の距離は指数関数的には広がり得ない。散乱が無相関である極限では粒子間の距離は拡散的に広がることになる。

そういうわけで、指数関数的に軌道が広がるといっても、それはせいぜい $R/\sqrt{N}$ 程度の距離までである。そこから先は拡散的なので、平均的には $\sqrt{t}$ に比例して距離が広がっていくということになる。

もっとも、これは依然として「粒子の位置がどこかわからなくなる」ということを意味することに変わりはない。拡散的になってしまうというのは、要するに本当にランダムになってしまうということである。もちろん、原理的には決定論的なハミルトン系なのでランダム性なんてものはどこにもないが、しかし、非常に近くにある 2 粒子の距離が、まずは指数関数的に、それからはランダム・ウォーク的に広がるわけである。繰り返しになるが、これは物理的な系のもともと持っている性質であり、数値計算で有限の精度でやるからそうなるとかいう問題ではない。

実際、系が熱力学的に振舞うことを保証しているのは、このように粒子の軌道が拡散的に広がっていくという性質そのものであるはずである。近くの軌道が指数関数的に広がる速さは熱力学的な振舞いに直接つながるわけではない。

6.5.3 なにを「正しく」できるか

では結局数値計算で正しいものは何で、それはどうすれば正しいということがわかるのであろうか?正しいものはなにかというと、結局、個々の粒子の位置には左右されないような統計的な性質であるというのがまあ優等生的な答ではある。具体的には、前のほうで述べた重力熱力学的カタストロフィーや自己相似的進化がそのような系の統計的振舞いであると我々が理解できるものの例ということになる。これらは、等質量の質点系で球対称という極めて単純な系の場合に起きることであり、これらの条件を外した系ではもっといろいろなことが起きるわけだが、そういうものについてもそれが統計的な性質である限りは「正しい」ということになる。

とはいえ、なにが統計的な性質でなにがそうでないかというのはそんなに自明ではない。この一つの理由は、実際の系でも系を構成している粒子の数がそれほど多くはないことであり、もう一つは、例えば自己相似的な進化自体の結果として粒子数が小さなサブシステムが勝手にできてきて、それがどうなるかが系の進化を決めることになるからである。

例えば、以下のような例を考えてみよう。前に重力熱力学的振動について述べたが、ここで、収縮および膨張自体をドライブするのは重力熱力学的不安定性であり、これら自体はもちろん熱力学的な性質から出てくるものである。これに対し、収縮から膨張へのスイッチングを起こすのは連星系からのエネルギー供給である。

連星系が効率的に形成されるためには、3 体衝突が起きる必要があり、そのためには系の速度分散で正規化した密度が非常に大きい必要がある。このためには、コアの粒子数が小さ

くないといけない。というか、そもそもこのような系の粒子数に対する依存性が連星系の生成確率にあるので、中心コアが小さくなると連星系からのエネルギー供給が無視できなくなってくるわけである。ここで、実際に連星系ができるようになってくるのはコアの粒子数が 100 以下になってきたところであり、同時に存在する連星は数個程度である。このために、連星系からのエネルギー供給には非常に大きな揺らぎが存在する。

ここで、エネルギー供給に揺らぎがあっても膨張・収縮は熱力学的に起こるのであるから、多少の揺らぎはそのなかで吸収されると想像するかもしれない。しかし、話はそんなに甘くはなくて、実はこの重力熱力学的振動が、エネルギー供給が決定論的、つまり、密度の 2 乗といった形で与えられ、さらに粒子数無限大の連続極限をとった場合でも、安定なリミットサイクルを持つものではなくカオス的になるということが、フォッカープランク計算に人為的にエネルギー入力を入れたシミュレーションで示されている[3]。つまり、多体計算の場合には、もともと決定論的カオスであるようなものにさらに粒子数が小さいことによる揺らぎが入っているわけである。このような時に、どのようにして計算結果が正しいといえるかというのは難しい。

例えば、計算精度を変えて同じような答がでるかというわけだが、粒子数が小さいための揺らぎが系全体の振舞いを変えるので、例えば振動の位相といったものは計算精度を変えれば全く変わってくる。多体系はカオス的で一つの粒子の軌道には意味がないという話をしたが、このように一つの粒子の軌道が系全体の振舞いに影響を与えるような場合にはそれでは何が正しいといえるのかということが問題なわけである。

もっとも、これはもともと自由度が小さくてカオス的な系と同じ問題であり、特に多体系の固有の問題というわけではない。結局、この場合にはさらにアンサンブル平均なり時間平均の振舞いが問題であるということになるが、カオス的であるとそういうものがちゃんとあるかどうかも自明ではない。

まあ、そういうわけで、なにが正しくて、どうすれば正しいといえるかというのは、結局のところ良くわからない。普通どうやっているかというと、例えばエネルギー保存を見てシミュレーションの 10^{-4} とかそんな程度で保存していればまあ良かろうとするわけである。これが 10% とかいうとまあ計算結果を信用する気が起きない。では、1% とか 0.1% ではいかんのかというと、これはなかなかなんともいえない。例えば、重力熱力学的カタストロフィーによる自己相似的な中心密度の上昇を再現するだけなら、エネルギーの誤差が例えば 0.1% と大きくてもあまり問題ではないようである。

実際問題として、エネルギーを見るというのは、特に、積分スキーム自体がエネルギー保存を保証しないような場合には、かなり厳しい条件になっていて大抵うまくいく。もっとも、後述のシンプレクティック公式や時間対称型公式のように原理的にはエネルギーの誤差が積もっていかない方法の場合には、エネルギー保存をみていていいのかどうかというのは明らかではない。

---

[3] 実際にはエネルギー供給の大きさによって振舞いが変わり、大きいと安定な膨張モードがあるが、これを小さくしていくと周期的な振動モードから 2 重周期、4 重周期と分裂して最終的にカオス的になる

6.5.4 時間刻みの決定

だいぶ話が拡散したが、ここでのもともとの議論はどうやって時間刻みを決めるかということであった。

いずれにしても軌道の時間積分の誤差をみてそれを評価したい。実用上の観点からすると、なんらかの意味で精度を与えるパラメータが相対的な形になっている、つまり、例えば単位系をとり直しても同じパラメータが使えることが望ましい。これは、極めて実際的な問題としては、系の大きさや粒子数を変えるたびに精度パラメータを考え直さないといけないのは面倒であり、間違いのもとになるからである。また、1 つのシミュレーションの中で、系が全体として膨張していくような場合に、それに応じて精度パラメータを変更するのは繁雑に過ぎる。

一方、ガリレイ変換に対して不変ではないという明らかな問題から、位置や速度の値自体を相対誤差の指標として用いることはできない。重力ポテンシャルは原理的には利用できるが、例えば銀河団の中の銀河の中にある粒子といったものを考えると重力ポテンシャルは銀河団全体が作っているものが大きいので、銀河団の中のどこに銀河がいるかで時間刻みが変わるという妙なことになってしまう。

そこで、重力加速度と、その高次の時間導関数だけを使って構成され、時間の次元を持つような量が誤差の指標としては望ましい。予測子と修正子の差の絶対値を加速度の絶対値で割ったもの

$$\left(\frac{|\boldsymbol{x}_p - \boldsymbol{x}_c|}{\Delta t^2 \boldsymbol{a}}\right)^{1/(p-1)} \tag{97}$$

は一応そのような性質を持っている。ただし、たまたま加速度が 0 というような場合には妙なことになるので、実際に使う時には注意が必要である。

式 (96) はそのような、加速度が 0 の場合でも大丈夫な形になっている。その代わり、明示的に誤差を見ているわけではない。これは、結局、積分スキームの誤差を評価してそれを一定に保つというよりは、むしろ単に高階導関数の値を見て、補間多項式に対する寄与があまり大きくなり過ぎないように適当に時間刻みを小さくするというようなものである。

例えばエネルギー保存を見た時に、この式は比較的うまく働き、例えば予測子と修正子の差を誤差とするようなやり方とほぼ同等な性能を出す。つまり、時間ステップの数が同じ時の誤差が同程度である。

もっとも、積分公式として 4 次以上のものを使った場合には、当然ではあるが式 (96) ではうまくいかない。ここでうまくいかないとは、パラメータを変えてタイムステップを小さくした時に、積分公式の次数から期待されるような割合でエネルギーの誤差が小さくならないということである。この場合にはもちろんより高次の導関数まで入ったものを使うべきであり、そのためには式 (97) のような、誤差を明示的に評価するものをつかうのが容易であることはいうまでもない。

6.6 特異点と正則化

「特異点」とかいうとなんかかっこいい話みたいに聞こえるが、ここではなんということはなくて重力ポテンシャルは距離の逆数つまり $1/r$ で、$r \to 0$ で発散するというだけのことである。この発散するポテンシャルがさらに引力であり、しかも古典力学系であるという事実のために、質点系のシミュレーションでは 2 粒子間のポテンシャルが実際に発散し得る。これに対する対策には基本的に 2 つある。一つは、多少物理に影響するかもしれないがポテンシャルをいじって発散をなくしてしまうというものである。もう一つは、真正直に $1/r$ ポテンシャルを使い、数値計算法を工夫することで対応しようというものである。以下、順に解説しよう。

6.6.1 ポテンシャル・ソフトニング

これは、つまり、r が小さいところで重力の形を変えてしまって、発散が起きないようにしようというものである。問題は、こういうことをすると系の進化を決める物理が変わってしまわないかということであるが、実は粒子数 N がある程度大きいところではあまり物理は変わらない。

ここで、物理といっているのは 2 体緩和である。2 体緩和は、前に説明したように近接散乱よりも主に遠くの粒子との相互作用が効いている。特に、距離 N/R 程度以下、つまり、散乱角が 90 度程度、あるいはそれ以上になるような近接散乱は、2 体緩和にはほとんど寄与していない。従って、その辺の散乱を精密に追いかけなくても、系の進化にはほとんど影響はないと考えられる。

また、問題によってはそもそも 2 体緩和自体正しく表現する必要はない、あるいはむしろ積極的に押えられるものならば押えたいことがある。これは、例えば銀河系のシミュレーションのように、現実の系では N が非常に大きくて 2 体緩和のタイムスケールが非常に長いのに、シミュレーションではそれほど大きな粒子数が使えないので 2 体緩和が起こってしまうというような場合である。

これらの場合には、r が小さいところで発散しないようにポテンシャルをなまらせることができる。なまらせる方法はいろいろあるが、もっとも広く使われているのは

$$\phi_{\rm soft} = -\frac{1}{\sqrt{r^2+\epsilon^2}} \tag{98}$$

というふうにする方法である。なお、文献ではこれはプラマー (Plummer) ソフトニングと呼ばれることが多い。これは、プラマーモデルと呼ばれる自己重力多体系の力学平衡分布モデルのポテンシャルがこの形になっているからである。このモデルは、銀河や星団のなかの星の分布のモデルとしてそんなにいいわけではないが、分布関数や密度分布等が単純な初等関数で書けるという特色がある。そのため、それほど正確な形が必要というわけではない時には銀河や星団のモデルとして良く使われる。

歴史的には銀河団の進化のシミュレーションで、銀河をこの形のポテンシャルで相互作用する剛体粒子（といっても、重力だけで相互作用するので、相互にかさなりあうことができる）で表現した [Aar63] のがこの形が使われた最初であるらしい。といっても、銀河団の

シミュレーションをやろうと思ってこの形を使ったのか、この形を使った計算しかできなかったからやった計算を「銀河団のシミュレーション」と称したのかというとどうも後者のように思われる。

と、余談はさておき、式 (98) の形のソフトニングのいいところは、非常にプログラミングが楽で、計算コストも低いことである。つまり、あらかじめ ϵ^2 を適当な変数にもっていれば、加算一つだけでソフトニングが実現できる。さらに、非常に滑らかな関数でソフトニングしたあとは全領域で無限回微分可能であり、高次の時間積分法等を使っても問題が起きないということもある。

もちろん、滑らかであるということは欠点と背中合わせでもある。つまり、厳密な $1/r$ ポテンシャルからのずれが r を大きくしてもゆっくりとしか小さくならないのである。式 (98) からすぐにわかるように、$r >> \epsilon$ ではポテンシャルや重力の相対誤差は ϵ^2/r^2 の程度であり、r の 2 乗でしか落ちない。これはなんとなく嬉しくないような気がするわけで、もっと高次のベキで落ちるとか、あるいは有限距離で誤差が 0 になるようなソフトニングのやりかたがいろいろ提案されている。

その中で、代表的なものはスプラインソフトニング [HK89] と呼ばれるものである。といっても、ポテンシャル自体をスプライン関数で表現するわけではなく、粒子の半径方向の密度分布をスプライン曲線で表現されるとして、それを積分した形のポテンシャルを使う。ある半径 $2h$ の外側では密度が 0 になるように作るので、そこでは厳密な $1/r$ ポテンシャルと一致することになる。

といっても、r の 2 乗でしか落ちないことが本当に問題かどうかはあまり自明な問題ではない。例えば実際にエネルギーの誤差を計算すれば、これは、通常のプラマーソフトニングではϵ^2のオーダーになっている。ではスプラインソフトニングではどうなるかというと、これも実は h^2 のオーダーになって大した違いはない。ただし、係数はスプラインのほうが多少小さい。

係数が小さいということは、誤差を同じにするなら h の値を ϵ の値に比べて多少大きくできるということであり、これは後で述べるようにタイムステップを多少大きくできる可能性があるということでもある。が、スプラインの場合には関数の連続性に制限がつくわけであり、これが時間積分にどのように影響するかはよくわかっていない。このため、どのようなソフトニングが「最良」であるかはまだ未解決の問題である。

なお、ソフトニングがある程度大きければ、タイムステップに実質的に下限ができるということには注意する必要がある。今、ソフトニングが R/N よりも十分に大きいとしよう。この場合、2 体の近接散乱で、粒子が曲がる角度は非常に小さくなる。また、散乱が起こっているあいだも速度はほとんど一定とみなせる。

この時明らかなことは、タイムステップを ϵ/v の数分の 1 程度にとってもよいということである。特にタイムステップが一定ならばこれで大きな問題はない。ここで v は典型的な粒子間の相対速度である。

さて、これよりもずっとタイムステップを大きくとってもあまり問題はおきないということが実験的には知られている。つまり、ある程度粒子数が大きく、またソフトニングも大きければ、系のポテンシャルは軌道の数値積分の精度という観点から見る限りにおいて実

効的になめらかになる。このために時間ステップはその滑らかな軌道を積分するために必要な程度でよく、ソフトニングの大きさとは無関係になるわけである。

もうちょっと正確にいうと、実際に滑らかになるわけではないが、タイムステップが大きいために高周波成分をちゃんと積分できてない（いわゆるエリアシングエラーが入っている）分が無視できる程度に小さくなるということである。無視できるための条件は、結局曲がる角度が小さいということである。

もちろん、この場合、エリアシングエラーによって高周波成分の寄与は増幅されていることには注意する必要がある。この増幅のために、各粒子のエネルギーは 2 体緩和で変わる分以上に変化している。これが計算結果に影響していないかどうかは確認しておく必要がある。

さらに、このようにタイムステップを大きくとれるのは、タイムステップを一定にとった場合だけであることにも注意しておく。タイムステップを適応的に変える場合には、結局高周波成分を拾ってそれに応じてタイムステップが変化する必要がある。そうなっていないと、そもそもまともにタイムステップを変化させられないからである。このために、タイムステップは通常ソフトニングサイズを分解できる程度に小さくならざるをえない。

このために、比較的大きなソフトニングを使った計算では時間刻みを下手に変えるよりは一定にしたほうが結局良い結果がえられる。この場合に時間刻みをどのようにとればいいかもまた未解決の問題である。

6.6.2 正則化

ソフトニングのところでは、$1/r$ ポテンシャルのために起きる発散をそもそも起きないようにするという話をしたが、どんな問題でもそういう扱いをしていいというわけではもちろんない。例えば、すでに述べたように理想化された質点系では重力熱力学的カタスロフィーの結果自己相似的に中心密度が増加する。3 体衝突で連星ができることによってこの自己相似的な成長は止められるわけだが、ソフトニングがあるとそもそも連星ができなくなってしまう。というのは、連星が回りにエネルギーを有効に与えるためには、その「温度」が回りの「温度」よりも高い必要がある、つまり、大雑把にいって連星内の軌道運動の速度が回りの星の速度分散より大きくないといけない。これは、軌道半径が R/N よりも小さいということとほぼ同一になる。従って、ソフトニングがこれよりも大きいと連星ができないことになる。

このような場合には、どうしても 2 粒子が非常に近付いた時でもうまく計算できるような方法が必要になる。ポテンシャル、加速度が発散するので、これは一見原理的に不可能なような気もするが、実用的にはなんとかなる。というのは、2 体が十分近付けば、その 2 体の相対位置については 2 体問題の解析解なり、あるいはもっと高い精度が必要であれば回りの粒子からの摂動を入れた解を使えばいいからである。近付いた粒子はいずれ離れるので近い時だけ解析解を使えばポテンシャルが発散しても問題ではない。

もちろん、このような扱いをするためにはまずその 2 体を認識する必要があるが、これは単にある適当に設定した限界距離以下になるという条件でいいのでそれほど面倒な話では

ない。

つまり、具体的には以下のようなことをする手続き群が必要になる。

- 2 つの粒子が近付いたかどうかをチェックする
- 近付いていたら、その 2 粒子の重心および相対位置（内部運動）を新しい変数として時間積分する
- 相対運動に対して、回りの粒子からの摂動が十分に小さければ数値積分しないで解析的に解く
- 2 つの粒子が十分離れたらその重心と相対位置から元の粒子を復元する

さて、このセクションの表題であったところの、正則化という話はではどこにいってしまったのかというわけだが、これは以下のような話になる。

正則化の基本的な考えは、座標と時間を適当に変換して数値解の性質をよくしようというものである。実際には 3 次元の運動だが、ここでは 2 次元の例で説明する。2 次元座標 $\mathbf{r} = (x, y)$ を、

$$x = u_1^2 - u_2^2 \tag{99}$$

$$y = 2u_1u_2 \tag{100}$$

という関係を満たす新しい変数 $\mathbf{u} = (u_1, u_2)$ に変換し、さらに独立変数を時間 t から $rds = dt$ という変数変換で得られる変数 s に置き換える。すると 2 体問題の運動方程式

$$\frac{d^2\mathbf{r}}{dt^2} = -\frac{\mathbf{r}}{r^3} \tag{101}$$

が単振動の方程式

$$\frac{d^2\mathbf{u}}{ds^2} = \frac{E}{2}\mathbf{u} \tag{102}$$

に変換される。ここで E はシステムの全エネルギーである。単振動の方程式の方が 2 体問題よりも数値的な性質がよく、少ないステップ数（大きなタイムステップ）で精度よく積分できる。摂動も変換して扱えるので、この方法では軌道離心率が 1 に近い、あるいは厳密に 1 であるような場合でも摂動がある解を数値的に求めることができ、摂動が無視できるようなところは解析解でつなぐといった余計なプログラミング上の手間は必要でなくなる。

2 次元の場合にこの変換を提案したのは、Levi Civita[Lev06] である。これは単純で 2 次元の場合には極めて有効であるが、例えば星団の中の連星の軌道に使うには 2 次元では不便である。3 次元の場合に拡張することに初めて成功したのは Kustaanheimo and Stiefel[KS65] であり、Levi Civita から半世紀以上立った後のことである。2 次元の場合の本質は、位置ベクトルを複素数とみなしてその平方根をとるということであるが、3 次元ベクトルで平方根というのはうまくいかない。Kustaanheimo and Stiefel はスピノルを使って 4 次元に軌道を拡張することにより、実質的に平方根演算になるような変換を定義することに成功した。

4 次元で平方根というと、いわゆるハミルトンの 4 元数が使えないかと思う人もいるかも知れない。これはかなり最近まで良くわかっていなかったというか、あまり誰も真剣に考え

ていなかっただけかも知れないが、とにかくそういう状況であったが、最近になって可能であるということが示されている。

Kustaanheimo と Stiefel のもともとのものと 4 元数を使った定式化のどちらの場合でも、3 次元が 4 次元になったぶん自由度が冗長になっているが、これは適当に処理して問題ない。

このような正則化を行なうことで、ポテンシャルが発散していても計算精度としては問題なく扱えるようになる。「計算精度としては」と断ったのは、実はこれでは計算時間の方に問題が発生するからである。2 つの粒子が遠くからたまたま近付いて、双曲軌道を描いてまた離れていくようなケースでは、上の扱いで十分である。問題なのは、連星である。

繰り返し述べたように、連星は星団の中心核の密度が上がり、3 体衝突が起きるようになってくると形成される。さらに、一度形成されるとこれは他の粒子と衝突した時に、内部の軌道運動から相手および自分の重心運動にエネルギーを与える傾向がある。

なぜ一方的にエネルギーを与えることになるかというと、連星はエネルギーを与えた結果軌道長半径が小さくなり、その結果内部運動の速度がいっそう大きくなるからである。このこと自体が自己重力系の熱力学的な不安定性の一つのあらわれではある。

軌道長半径が小さくなるということはもちろん軌道周期が短くなるということである。どの程度短くなるかというと、おおよそ他の粒子が星団内を動くタイムスケールの $1/1000N$ 程度まで小さくなる。この程度で止まるのは、このあたりで次に他の星と衝突した時に連星の重心運動がもらうエネルギーが、星団を脱出できる程度まで大きくなるからである。

軌道周期が $1/1000N$ ということは、もちろん時間刻みがその程度短くないといけないということである。従って、独立時間刻みでやったとして他の粒子全部をあわせたのの 1000 倍の計算時間がかかるということになり、そういうものがごくわずかの時間でもできれば計算は全く進まなくなる。さらに、このような非常にコンパクトな連星ができると、これに他の星が近付いて共鳴状態になったものの計算ではタイムスケールが短いだけでなく、座標の桁落ちが無視できなくなる。

実際には、いくつかの細かい工夫の積み重ねでこれらの問題を回避している。

まず、十分コンパクトな連星で、回りの粒子からの摂動が無視できるほど小さい場合には、要するに単に無視すればよいわけである。この時は、連星の内部運動はケプラー運動であり、解析解がわかっているので全く計算する必要がなくなる。

ただし、どの程度摂動が小さくなれば無視してよいかというのは理論的には微妙な問題がある。というのは、軌道要素によって変化の摂動の大きさへの依存性がちがうからである。もっとも鈍感なのはエネルギー（軌道長半径）であり、これは断熱不変量なので摂動が小さくなる（変化が遅くなる）と指数関数的に変化が小さくなる。

質点系で、連星が主に熱源として重要であるという場合には、熱源としての働き（散乱断面積）にはエネルギー以外の軌道要素、つまり離心率やルンゲ・レンツベクトル（重心から近点に向いたベクトル）の影響はあまり大きくないのでエネルギーさえちゃんと計算できればあとはどうでもいい。これに対して、現実的な効果、例えば星が有限の大きさを持ち、潮汐力によって変形したり振動モードが励起されたり、あるいは重力波放射したりすることを考えにいれたいとすると、離心率が「正しく」計算されていることは極めて重要である。しかし、離心率の変化は摂動のベキでしか小さくならないので、これをきちんと

扱うためにはかなり摂動が小さいところまで無視できないことになる。

他の軌道要素まで正しくしようとするとさらに大変な話になるが、そこまでやる意味はあまりないと普通は考えている。というのは、ここでの「正しく」というのは所詮統計的な意味でしかないし、これは全体の N 体計算自体がそうであるからである。従って、ルンゲ・レンツベクトルや軌道の位相を正しく計算しても、回りの星の軌道がそれに見合う精度では計算できていないのであまり意味がないと考えられる。

さて、全く摂動が無視できるほど回りの星が離れてはいないが、といってまともに計算するのは大変過ぎるという領域では、摂動論的な扱いをしたくなる。もちろん、どうやるかは天体力学の教科書をみれば書いてあるわけで、そういうものを実装すればいいが、これは結構面倒臭い。というのは、摂動項を求めるには連星の軌道に沿って外力を時間平均して、それから各軌道要素の時間変化率を求めるという手続きが必要になるからである。さらに、軌道要素とデカルト座標なり正則化座標系との相互変換をする関数も必要になる。

実効的に摂動論的計算をするのに、このような面倒な手間を一切かけないですます画期的に安直な方法が、Mikkola と Aarseth[MA96] によって提案された。この方法の基本的な考えは、「摂動論的扱いができる範囲内で連星の軌道周期を遅くする」というものである。摂動項は、実時間での軌道要素の変化率が同じになるようにスケールして運動方程式に入れる。これによって、摂動論の範囲内では摂動論と同じ結果を、軌道を直接数値積分して得ることができる。これはコロンブスの卵的であるが、極めて有効な方法である。

もちろん、こういった方法ではうまくいかない場合というものが実は存在する。一つは、階層的な 3 重星、つまり連星の外側をもう一つ星が回っているようなものである。この場合には、もちろん安定であるとわかっていれば計算しなくていいわけだが、安定性条件は数値実験によっていくつかの場合について求められているだけで、N 体計算で形成されるさまざまな場合について機械的に安定性を判定する手法は存在しない。また、上に述べたようなタイムスケールと摂動項のスケーリングは、3 重星の場合には使えない。さらに、数値的に求まっている安定条件にしたところで、あくまでも「その時間まで計算した範囲では壊れなかった」というだけで、もっと長時間について安定かどうかはわからない。これは、太陽系が安定かどうかというラプラスの昔からあってまだ完全には決着がついていない問題と本質的には同じ話である。

ちょっと話題がそれるが、太陽系の安定性についての最近の研究について触れておこう。そのへんの本を見るとラプラスが太陽系の安定性を証明したとか書いてあるかもしれないが、現代的な理解ではこれは正しくない。この辺の詳しいことはウソを書いてはいけないので省略するが、最近の研究のハイライトは MIT のグループが摂動展開をしないで直接に外側 5 惑星の軌道の数値積分を行ない、リヤプノフ指数が正であることを発見した [SW88] ということである。驚くべきことに、リヤプノフ指数はその逆数がわずか 2 千万年と、太陽系の年齢であるおよそ 50 億年に比べると極めて短いものであった。

実際の太陽系の年齢がもっと短いというのはありそうな話ではないので、リヤプノフ指数から出てくる不安定のタイムスケールがこれほど短いというのは当初はなかなか信用されなかった。信用されなかった理由の一つは、MIT グループの使っていた計算法がちょっと怪しいもので、精度が十分であるかどうかはっきりしなかったことと、専用計算機 [ADG+85] を使った大計算で追試が困難であったことである。

90 年代に入ると、汎用計算機の高速化と次節で述べるシンプレクティック法など数値計算法の発展の結果、追試が可能になってきた。国立天文台の中井 ·木下らの追試 [KN96] では、確かにリヤプノフ指数は大きいものの、太陽系は 50 億年という長期に渡って安定であるという結果になっている。つまり、リヤプノフ指数が単純には系の不安定性の時間尺度を与えていないのである。さらに最近の結果では、500 億年に渡って安定という結果も得られている。現実問題としては太陽の寿命は 100 億年程度なので 500 億年という数字自体に意味があるわけではないが、いずれにしてもリヤプノフ指数から決まるタイムスケールよりもはるかに長い時間に渡って安定であるという結果になっている。

と、そういうわけで、階層的な 3 重星の扱いには未解決というか解決があるかどうかもわからない問題がある。これは、数値計算法の問題というよりは、むしろカオス力学の問題であるといえよう。つまり、最終的に安定かどうかがわからないと、どのように扱うべきかが決まらないのである。

さて、究極的に安定かどうかはともかく、階層的な 3 重星を安定に数値積分しようと思えば 3 重星の内部運動を内側の連星と外側の連星にわける、つまりツリー構造的な扱いをしたい。このような扱いをした場合、各自由度の運動方程式や、摂動項がどのようになるかは、再帰的な形に書き下すことで比較的簡潔に表現できる。[MT98, PMHM01] 再帰的な扱いを可能にしておくことで、3 重星に限らず 2 つの連星からなる連星や、もっと粒子数が多くて複雑なシステムも自動的に扱えるようになる。このように階層的にした場合には、各自由度は摂動つきのケプラー運動になるので KS による正則化をするなりあるいは古典的な方法で積分するなり、どんな方法でも使うことができる。

ややこしそうな気がするのは、どのような条件でツリー構造を新しく作ったり壊したりするかということだが、これは局所的に粒子 1 つだけをとったりつけたりすることで、ある瞬間をみたときに常に最適な構造になっているとは限らないが、数値的に問題を起こさない程度にはよい構造を実現できることがこれまでの実績からはわかっている。

階層的でない状態にある 3 重星や、4 個以上の系には、従来 General N-body relaxation [Heg74] や、その簡略版である Chain regularization[MA93b] と呼ばれる方法が使われてきた。General N-body relaxation は、簡単にいってしまうと n 個の粒子からなる系を考えた時に、 $n(n-1)/2$ 個のすべてのペアの相対距離に対して全部 KS 正則化をかけるような方法である。このため、方程式系は非常に冗長度が高く、計算コストが増えるが、そのかわり座標系の切替えをまったく必要としないで任意の 2 体の衝突を正則化できる。このために、切替えによる丸め誤差の累積を避けることができるという潜在的な利点があることになる。

筆者個人の意見としては、しかし、どのような方法を使ったところである程度のヒステリシスを設けることで、座標系の切替の回数は実用上問題にならない程度には小さくできるので、計算コストが増えることを補えるだけのメリットがこれらの方法にあるとは思えない。もしも切替えによる計算精度の低下が本当に問題になるなら、4 倍精度演算を使う等して精度が落ちないようにするのは容易だからである。

6.7 シンプレクティック型公式と対称型公式

さて、ここまで、数値積分自体に使う公式としては基本的にかなり古典的な PC 法だけを扱ってきた。PC 法は、すでに紹介したように古い歴史を持つ計算法であり、実際その歴史は電子計算機の発明以前に遡るものである。もちろん近年になっていろいろな発展がなかったわけではないが、我々が使っている方法は実際に 19 世紀末にアダムス等が定式化したものから大きく変わっているわけではない。アダムス法にせよ、それを 2 階の方程式に特化したシュテルマー・カウエル法にせよ、もとの微分方程式の局所的な近似解を基本的にはテイラー級数展開のようなものに基づいて構成するというだけのものであり、それ以上に元の系の力学的な性質を利用してよい結果を得ようというものではない。このような公式をハミルトン力学系である重力多体系に適用したときには、系のエネルギーのような保存量には通常時間に比例する誤差が現れる。

周期解をもつ調和振動子やケプラー問題ではエネルギーのような保存量に時間に比例する誤差が現れるのはまあ当然のことであろう。誤差が小さい、線形の範囲であれば、最初の 1 周期に出た誤差と同じような誤差が繰り返しでるからである。多体系の場合には各粒子の軌道はカオス的であるとはいえ、平均的には他の粒子全体が作る滑らかなポテンシャルのなかで準周期的な運動をするし、時々ある他の粒子との双曲的な接近遭遇では、積分スキームや時間刻みのとり方を決めればいつでも誤差の方向は同じになるので、やはり単位時間当たりの誤差はおおむね一定になる傾向がある。

多体系の場合にはどうせカオス的だし、エネルギーが保存しないからといってどれほど悪いかというのはそれ自体がよくわからない問題であるのはすでに述べた通りだが、前に述べた弱い摂動を受けた連星系を長時間積分するような場合だとエネルギーや角運動量が保存しないのは嬉しくない。というのは、それらさえ保存すればあとはどうでもいいというところがなくはないからである。従って、そのような「良い」公式はないかというのが実用上の問題になる。

経験的には非常に古くから、低次であるにもかかわらず特別によい振舞いをする公式があるということは、重力多体系、分子動力学計算、プラズマ物理などハミルトン系の数値計算が使われるどの分野でも知られていた。それぞれ別の名前がついており、また分野によっては（丸め誤差の範囲で）同じものがいくつかの名前で呼ばれているが、実体はみな同じものでいわゆるリープフロッグ公式である。(Verlet, 2nd order ABM, 2nd order Stömer-Cowell あたりが古くからある名前の代表的なものである) 一つの形は以下のようなものである。

$$v_{i+1/2} = v_{i-1/2} + \Delta t a(x_i) \tag{103}$$

$$x_{i+1} = x_i + \Delta t v_{i+1/2} \tag{104}$$

ここで、添字はステップである。 $\pm 1/2$ は中間点での値ということになる。これでは速度と位置がずれた時間でしか定義されないが、出発用公式として

$$v_{1/2} = v + \Delta t a(x_0)/2 \tag{105}$$

を使い、さらに終了用公式として

$$v_i = v_{i-1/2} + \Delta t a(x_i)/2 \tag{106}$$

を使うことで最初と最後を合わせることが出来る。この形は、実は

$$\begin{aligned} x_{i+1} &= x_i + \Delta t v_i + \Delta t^2 a(x_i)/2 \qquad (107)\\ v_{i+1} &= v_i + \Delta t[a(x_i) + a(x_{i+1})]/2 \qquad (108) \end{aligned}$$

と数学的に等価である。こう書くと PC 法みたいに見える。また、以下のようにも書ける

$$\begin{aligned} v_{i+1/2} &= v_i + \Delta t a(x_i)/2 \qquad (109)\\ x_{i+1} &= x_i + \Delta t v_{i+1/2} \qquad (110)\\ v_{i+1} &= v_{i+1/2} + \Delta t a(x_{i+1})/2 \qquad (111) \end{aligned}$$

これは一見なんだか良くわからない。さらにまた、速度を消去して x_{i-1}, x_i, x_{i+1} の関係式の形で書いてあることもあるかもしれない。

この公式は局所誤差が $O(\Delta t^3)$、大域誤差が $O(\Delta t^2)$ である。つまり、普通にいう 2 次精度である。局所誤差という観点からはこれは決して良い公式というわけではないが、現実にはこの公式は非常に広く使われている。

これは、別にもっとよい方法を知らないからとかではなく、実はこの方法がいくつかの意味で非常に良い方法であるからである。いくつかの意味とは、例えばエネルギーや角運動量のような保存量が非常によく保存するということである。これらの保存量については多くの場合に誤差がある程度以上増えない。

この、ある程度以上誤差が増えないというのはめざましい性質である。通常の方法では、エネルギーの誤差は時間に比例して増えていく。従って、長時間計算をしようとすればそれだけ正確な計算をする必要がある。ところが、エネルギーの誤差は溜っていかないのならば、かならずしも精度を上げる必要はないとも考えられる。

もちろん、エネルギーが保存していればそれだけで計算が正しいということにはならない。が、理論的には、いくつかの重要な結果が得られている。まず、

1. リープフロッグはシンプレクティック公式 のもっとも簡単なものの一つである。
2. リープフロッグは対称型公式 のもっとも簡単なものの一つである。

シンプレクティック公式については、以下のようなことが知られている

1. シンプレクティック公式は、すくなくともある種のハミルトニアンに対して使った場合に、それに近い別のハミルトン系に対する厳密解を与えることがある。
2. 周期解を持つハミルトン系に対して使った場合に、どんな量でも誤差が最悪で時間に比例してしか増えない。
3. 時間刻を変えると上のようなことは成り立たなくなる

これに対し、 対称型公式については以下のようなことが知られている

1. 周期解を持つ時間対称な系に対して使った場合に、どんな量でも誤差が最悪で時間に比例してしか増えない。
2. 時間刻を変えてもうまくいくようにすることも出来る。

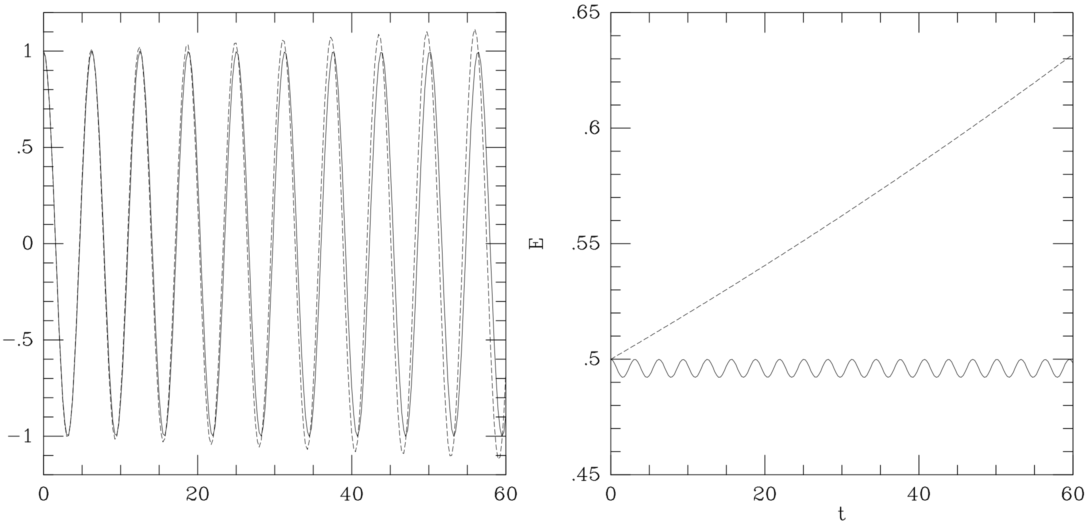

図 7: 調和振動子の数値積分。左 :軌道。右 :エネルギー。破線は 2 次のルンゲクッタ、実線はリープフロッグの結果

それぞれについての詳しい解説は、それだけでかなりの量になるので「数理科学」の吉田による解説 [Yos95] 等をみていただくことにして、ここではそれぞれの考え方に基づいてより高次の公式を構成するいくつかの方法について簡単にまとめる。

6.7.1 数値例

とはいえ、能書きを聞いていても良くわからないので、実例を見てみよう。まず、簡単な例として調和振動子

$$\frac{d^2x}{dt^2} = -x \tag{112}$$

を リープフロッグ と 2 階のルンゲクッタで解いた例を示す。初期条件は $(x, v) = (1, 0)$ で、時間刻みは $1/4$ である。

軌道とエネルギーを図に示す。非常に特徴的なのは、 リープフロッグ ではエネルギーが周期的にしか変化しないのに対し、ルンゲクッタでは単調に増えていることである。

ルンゲクッタでは単調に変化するというのは前に説明した通りである。エネルギーが増えるか減るかは自明ではなく、ルンゲクッタでもどんな公式を使っているかや時間刻みの大きさによるが、2 次の場合には公式や時間刻みによらずに必ずエネルギーが増える方向にいく。これに対し、リープフロッグではエネルギーが変化していない。これはどういうことなのであろう。

実は、この調和振動子の場合には、リープフロッグ公式は以下の量

$$H' = \frac{1}{2}(x^2 + v^2) - \frac{h^2}{4}x^2 \tag{113}$$

を保存するということを確かめることが出来る。つまり、 (x, v) で与えられる位相平面の

上で考えると、リープフロッグ公式の解は上の式で与えられる楕円の上にのっているのである。このために、エネルギーの誤差がある値よりも大きくなり得ないことになる。

では、非線形振動ではどうだろう?簡単な例として

$$\frac{d^2x}{dt^2} = -x^3 \tag{114}$$

をリープフロッグと 2 階のルンゲクッタで解いた例を示す。初期条件は $(x, v) = (1, 0)$ で、時間刻みは $1/8$ である。

軌道とエネルギーを図に示す。調和振動の場合と同様に、リープフロッグではエネルギーが周期的にしか変化しないのに対し、ルンゲクッタでは単調に増えている。

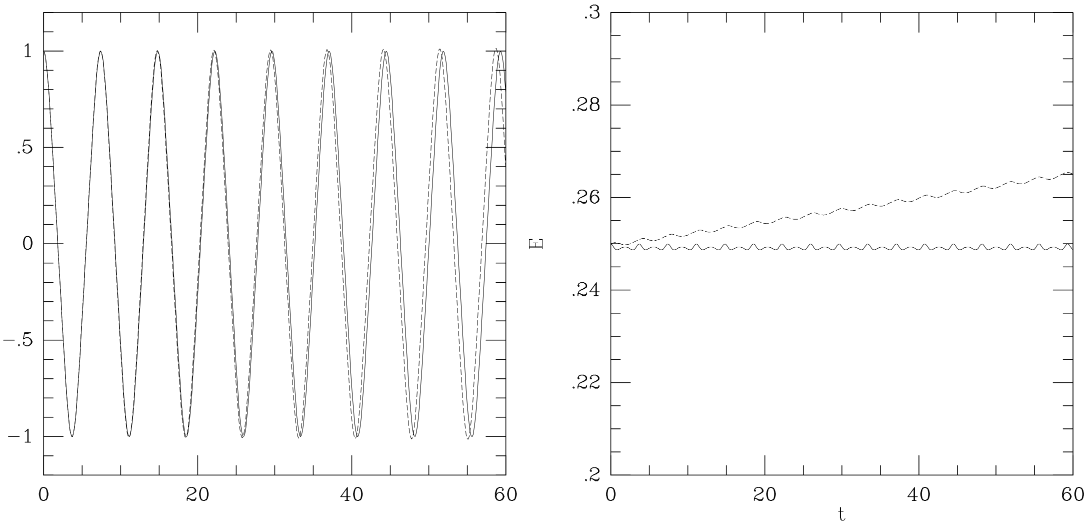

図 8: 非線形振動の数値積分。左 :軌道。右 :エネルギー。破線は 2 次のルンゲクッタ、実線はリープフロッグの結果

6.7.2　シンプレクティック公式

リープフロッグ公式は、すでに述べたようにハミルトン系に対してエネルギー誤差が有界に留まるという大きな特長がある。が、2 次精度でしかない。もっと高次の方法はないのだろうか、またあるとすればどういう原理で作れるのだろうか。

高次の方法を構成する一つのアプローチが、シンプレクティック公式といわれるものである。これはなにかというと、積分公式がシンプレクティック写像になるように作るということである。

シンプレクティック写像とは、要するに正準変換のことである。なぜそれではカノニカル公式といわないでシンプレクティック公式という難しげな言葉を使うのかは知らないので聞かないでほしい。正準変換とはなにかというのは、まあ直観的には力学系を不変に保つ、

つまり変換前の座標系で求めた軌道を変換したものと、変換後の座標系での力学系の軌道が同じになるようなものである。

ハミルトン力学系の解そのもの（ある時刻 t での座標から、$t+h$ での座標に移す変換）もシンプレクティックである。まあ、だから、シンプレクティックになっているような積分公式は、そうでないものに比べてなんとなく力学系の性質にあっているような気はする。

なお、リープフロッグがシンプレクティックであることを示すのは、これがある瞬間の加速度で速度だけを進める写像と、ある瞬間の速度で位置だけを進める写像であることに注意すると、それらがそれぞれシンプレクティックであることを示せばいいことになって結構簡単である。さらに、このことから、リープフロッグのようにこの 2 つを組み合わせて構成される写像、特にリープフロッグを刻み幅を変えて繰り返し適用する写像はすべてシンプレクティックであることになる。

で、いいたかったことは何かっていうと、上の リープフロッグ公式はこのシンプレクティック性を満たしているということだった。以下、高次の公式にはどんなものがあるかという話をしておく。

6.7.3　陽解法

陽解法の組み立て方は、基本的にはルンゲ・クッタ・ナイストレム型の公式、すなわち、2 階の方程式に特化したルンゲ・クッタ型の公式で、係数をシンプレクティック性を満たすように決めるということである。これはここ 10 年で無数に論文がでた。4 次、あるいは 6 次の公式としては、吉田や鈴木による作用素分解に基づく公式が良く知られていて、広く使われていると思う。

これらの方法の原理は、要するに先に述べたようにリープフロッグをタイムステップを変えていくつか組合わせるというものである。うまくタイムステップを組み合わせると誤差の高次の項を消すことができるわけである。7 段 6 次や 15 段 8 次の公式等が吉田によって導かれている。鈴木による再帰的な公式は、原理的にいくらでの高次のものが構成できるという意味では美しいが、指数関数的に段数が増えるので実用性からはちょっと疑問な気もする。

なお、実は陽解法はハミルトニアンが $T(p)+V(q)$ の形の場合にしか使えないが、大抵の問題はこう書けることはいうまでもないであろう。

また、RKN 系の公式を力任せに構成する試みもあり、4 次から 8 次までの公式が作られている。計算精度という観点からはこちらのほうがリープフロッグを組合わせるものよりも何桁も良いようである [SSC94]。

6.7.4　シンプレクティック公式の意味

さて、シンプレクティックであるということと、「良い」ということの間にはちゃんとした理論的な関係があるのであろうか？一応あるということになっている。つまり、あるハミルトニアン H で表される系をシンプレクティックな p 次の公式を使って積分した解は、別

のハミルトニアン H'で与えられるシステムの厳密解になっていて、H と H' の間に

$$H = H' + h^{p+1}H_p + O(h^{p+2}) \tag{115}$$

という関係がある（H'を求める数列が収束すれば）ということがわかっている。

なお、例えば上の調和振動子の場合には実際に数列が収束し、H' が求まっている。が、これは極めて例外的な場合で、一般の場合には収束するかどうかも明らかではないようである。

収束するかどうか明らかではないのでは、使っていいことがあるという保証はないではないかと思うかもしれない。理論的にはその通りなのであるが、実際にはいろいろな問題に適用されて、従来の方法よりも高い精度が得られるということが確認されている。

6.7.5 シンプレクティック公式の問題点と対応

さて、式(115)をみるとわかるように、シンプレクティック公式に付随するハミルトニアン H' には時間刻み h が入っている。従って、h をふらふら変えると H' も変わって、結果的に求まった数値解は一つの力学系の軌道ではない変なものになってしまう。ということは、可変時間刻みにするとシンプレクティック公式はうまく働かないのではないかということが想像される。

実はその通りで、例えばシンプレクティックで埋め込み型のルンゲクッタというものを作って、実際に時間刻みを変えてみた人がいる。その結果、普通のルンゲクッタよりも良くならないということが発見された [CSS93]。多くの問題でこれは致命的な欠陥となるので、さまざまな対応法が精力的に研究されているのが現状である。が、あまり一般的にうまくいく方法というのは見つかっていないようである。つまり、万能な公式というのはシンプレクティック公式に関する限り知られていない。

なお、シンプレクティック公式において普通に時間刻みを変えると、それはシンプレクティックではなくなるということが Skeel と Gear により一般に証明されている [SG92]。彼らの証明は、大雑把にいうと時間刻みも含めて考えると、写像が時間方向に歪んだもの（もとの位置によって時間刻みが変わるため）になり、そのために元の公式がシンプレクティックであれば時間刻みを変えると必ずシンプレクティックでなくなるというものである。

それでは、時間刻みを変えるのは絶対に不可能かというと、以下のような考えかたでの研究が進められている。ハミルトニアン H を

$$H = H_1 + H_2 + \cdots \tag{116}$$

と複数の項の和にわけ、それぞれについて違う時間刻みを与えることで実効的に時間刻みを変えるというものである。（例えば [SB94]）

上のようなハミルトニアンの書き換えは、自由度が小さい系ならできなくはないかもしれないがここでテーマにしている重力多体系では実際的ではない。というわけで、現在のところ、古典的なリープフロッグ公式がソフトポテンシャルと組み合わせて一定時間刻みで使われている他は、シンプレクティック公式がそのまま重力多体系に使われることはほとんどない。

もう一つのシンプレクティック公式の問題点は、「写像」であるということから一段法、具体的にはルンゲクッタ型の公式であるので、局所誤差に対する計算量という観点からは線形多段階法に比べて必ず悪いということである。実は線形多段階法を持ち出すまでもなく局所誤差に対して最適化したルンゲ・クッタ法に比べても、驚くほど誤差の係数は大きい。例えば 8 次の公式の場合、15 段の吉田による公式の誤差の係数は、2 階の問題に最適化された非シンプレクティックな Dormand らによる 9 段公式（実質 8 段 8 次）の 10^4 倍にもなる。ステップ当たりの計算量が約 2 倍であることを考慮すると、同じ計算時間での局所誤差は 100 万倍も大きいということになる。まあ、8 次の公式なので、精度を同じにするとして計算時間は 6 倍程度で済むが、よほど長時間の積分でなければ非シンプレクティックな公式のほうが実は良かったりする。4 次の場合も、同様に良い非シンプレクティックな公式と普通に使われているリープフロッグ 3 回の組合わせでは 2 桁ほど局所誤差に差がある。

なお、シンプレクティックであってなおかつ局所誤差が小さい公式というものも存在はしている。代表的なものは陰的ガウス公式である。これは、ガウス・ルジャンドル型の数値積分公式から導かれるいわゆる collocation 法といわれるクラスの陰的ルンゲクッタ公式である。ルンゲ・クッタなので積分区間の途中の何点かで関数の値を計算する。陰的ガウス公式ではこの点をルジャンドル多項式の 0 点にとり、計算した導関数値から補間多項式をつくってそれを区間で積分することで次のステップでの解を求める。さらに、collocation 法であるとは、途中の点で導関数値を計算するのにその点での近似解がいるが、その解に補間多項式を積分したものを使うことを指している。このためにこの方法では補間多項式の係数についての陰的な方程式を解く必要がある。

この方法の重要な利点は、段数に対して 2 倍の次数を達成できることである。つまり、4 点を使う公式で 8 次が実現できる。陽的な方法ではシンプレクティックでなくても 8 次には 9 段が最低限必要であり、それよりもだいぶ少ない。また、局所誤差の係数も極めて小さい。

しかしながら、陰的公式であり何らかの方法で方程式を解かないといけない。自由度が大きい時には何らかの方法といっても直接代入以外は現実的ではないので、まあまあよい初期推定をもとにすることで数回で反復を止められるようにしないと結局かえって遅くなることにはなる。とはいえ、原理的にはいくらでも高次の方法が機械的に構成でき、さらに計算コストも少なくなる可能性があるという意味で魅力的な方法である。単なる 2 体ケプラー問題で 8 次のシンプレクティック公式同士を比べた時には、直接代入で収束させたガウス公式がもっともよいという結果が得られている [SSC94]。

この方法のさらにもう一つの利点は、ハミルトニアンが分離可能でなくても使えるということである。

6.7.6 MVS 公式等

前節で、シンプレクティック公式がそのまま重力多体系に使われることはほとんどないと書いたが、これは「そのまま」でなければ使われている例 [WH91] もあるということでもある。ここではそれがどんな例か説明しておく。とりあえず、どういうものに使いたいかというと、例によって摂動を受けたケプラー問題である。この方法自体は、重力多体系といっても恒星系ではなく、太陽系の惑星の長時間積分のために開発されてきたものである

が、すでに述べた連星の扱い等にも応用されるようになってきているし、また他の分野への応用もひろいと考えられるので少し詳しく見ていくことにする。

リープフロッグと、その組合せからなるようなシンプレクティック公式の基本的な考えは、ハミルトニアン

$$H = T(p) + V(q) \tag{117}$$

に対して、p に対する以下の変換

$$p \leftarrow p - \Delta t \frac{\partial H}{\partial q} \tag{118}$$

と q に対する同様な変換がそれぞれシンプレクティックなので、それらを順番に適用したものもシンプレクティックになるということであった。

さて、摂動をうけたケプラー問題では、ハミルトニアンは

$$H = T(p) + V_1(q) + V_2(q) \tag{119}$$

と書ける。惑星系なら V_1 が太陽からの重力で、V_2 は自分以外の惑星からの寄与ということになる。このようにハミルトニアンを分解した時に、

$$H_1 = T(p) + V_1(q) \tag{120}$$

の解はそれ自体シンプレクティックであり、また V_2だけを考えた

$$p \leftarrow p - \Delta t \frac{\partial V_2}{\partial q} \tag{121}$$

というマッピングもシンプレクティックである。従って、この2つを組み合わせて積分公式をつくることができる。ここでやっていることは、結局普通のリープフロッグでは $x \leftarrow x + \Delta t v$ と直線で動かす代わりに、ポテンシャル V_1 に沿って動かすというだけである。ケプラー問題は解析的に解けるので、このようなやり方で太陽中心の軌道を極めて高精度に積分できることになる。V_2 の寄与については普通にやったのでは2次精度になるが、普通のリープフロッグと全く同様にステップサイズを変えたものを組み合わせることで高次にすることもできる。

実用的にこの方法を使うためには、ケプラー問題を高速に解く方法が必要になる。これは、実空間でまともに解く方法と、KS 変換した座標系で解く方法のそれぞれで極めて高速に解ける方法が提案されている。太陽系の惑星のように軌道離心率がそれほど大きくない場合には実空間で扱うのが簡単でいい。離心率が大きい場合にはKS 変換の利用も検討はしてみるべきであろう。

なお、このような「ハミルトニアンの分割」という見栄えのいい方法ではないが、太陽重力とそれ以外の惑星からの摂動を別に扱おうという考え方は天体力学では非常に古くからあり、エンケの方法として知られている。古くからあるだけになにをもってエンケの方法というかも本によって微妙に違うので、以下には最近の榎森ら [EIN93] による定式化の簡略版を述べる。

もともとの運動方程式を

$$\frac{d^2\mathbf{r}}{dt^2} = \mathbf{f}(\boldsymbol{r}) + \mathbf{F}(\mathbf{r}, t) \tag{122}$$

という形に書く。ここで $\mathbf{f}(\boldsymbol{r})$ は太陽からの重力であり、$\mathbf{F}(\mathbf{r}, t)$ は外力である。外力のない 2 体問題の解析解が $\mathrm{r}_0(t)$ であたえられる時、$\Delta\boldsymbol{r} = \boldsymbol{r} - \boldsymbol{r}_0$は以下の方程式に従うことになる

$$\frac{d^2\Delta\boldsymbol{r}}{dt^2} = \mathbf{F}(\mathbf{r}, t) + \mathbf{f}(\boldsymbol{r}) - \mathbf{f}(\boldsymbol{r}_0) \tag{123}$$

この方程式の右辺はルンゲ・クッタやエルミート法ではそのまま数値積分できる。線形多段階法の場合は少し複雑になるが、うまく使う方法はある。この方法では、数値積分する$\Delta\boldsymbol{r}$の大きさは、もともとの$\boldsymbol{r}$よりもずっと小さいので、それだけ計算精度があがる。特に、外力が非常に小さい時にはタイムステップを長くとれる。なお、2 次の MVS はこの方程式をリープフロッグで積分していることに相当する。

こちらの定式化の利点は、MVS に比べて高次で局所誤差の小さい方法が使えることである。そのかわり、摂動項に対してはシンプレクティックでなくなっているので、時間に比例する誤差が完全にはなくならない。

6.7.7　対称型公式

シンプレクティック公式は、とりあえずうまく動くのでいろいろな分野で使われるようになりつつあるが、欠点もいくつかあるのはすでにみた通りである。大雑把にいえば、シンプレクティック公式の利点はほぼそのままにした上で、欠点に対応できそうなのが対称型公式ということになる。

まず対称型公式とはなにかということだが、まずここでいう時間反転に対する対称性というものを定義しておこう。時間反転に対する対称性とは、常微分方程式

$$\frac{dy}{dx} = f(x, y) \tag{124}$$

に対する一段法

$$y_{i+1} = F(x_i, y_i, f, \Delta x) \tag{125}$$

が、

$$y_i = F(x_{i+1}, y_{i+1}, -f, \Delta x) \tag{126}$$

を満たすこと、つまり、直観的には、ある微分方程式系があって、それを 1 ステップ分数値的に軌道を進めたとする。で、そこから逆に戻ると厳密に元の値に戻るということである。(ここでは丸め誤差はないものとする)

で、一段法の場合には、この意味で対称なものを対称型公式ということにする。

具体例で考えてみよう。例えば前進オイラー法は対称型ではない。これは、いった先で導関数を計算すれば、一般にもとのところでの導関数とは違うから当然であろう。前進オイラー法が対称ではないのだから、その逆写像である後退オイラー法も対称型ではないことになる。

では、対称型にはどういうものがあるかということだが、明らかに対称型であるものとしては、台形公式

$$y_{i+1} = y_i + \frac{h}{2}(f_i + f_{i+1}) \tag{127}$$

がある。これが上の対称性を満たしていることはいうまでもない。

さて、なぜこの型の公式を考えるかということであるが、少なくとも実験的には以下のようなことが知られている。[HMM95]

- ハミルトン系に対して対称型公式を使った場合、エネルギーや角運動量などの保存量の誤差がある範囲内にとどまる
- 時間刻みを変えても上の性質を保つことが出来る

前のほうの性質はシンプレクティック公式と同じであるが、後の方の性質はシンプレクティック公式よりもある意味でよいものである。

さて、一階の方程式に対する一段法という制限をつけた場合、つまり、ルンゲクッタ法の場合、対称な公式はかならず陰的公式になる。逆に、陰的公式であれば対称型のものを作るのは容易である。具体的には、前に出てきた陰的ガウス法は対称型公式でもある。

一階の系に対する通常の一段法では、これよりよい方法は多分ない。それでは、

- ハミルトン系専用の解法ではどうだろうか
- 多段法ではどうなっているのだろう
- それ以外の方法はないのだろうか

以下、順番に考えていくことにする。

6.7.8　ハミルトン系用の陽的対称型 RK 公式

実は、すでに述べたように、リープフロッグ公式は対称型でありしかも陽公式である。で、リープフロッグの組合わせで作られる公式も、実はすべて対称型になっている。

6.7.9　対称型線形多段法

さて、一段法という枠を外してみよう。

まず、線形多段階法に対する対称性の定義を与える必要がある。ここでは、以下のように定義しておこう。時間刻みが一定の時、2 階の方程式用の線形多段階法は、一般に

$$\alpha_0 x_{i+p} + \cdots \alpha_p x_i = h^2(\beta_0 f_{i+p} + \cdots \beta_p f_i) \tag{128}$$

という形に書くことができる。 $\beta_0 = 0$ なら陽的であるし、そうでなければ陰的公式であ

る。ここで、対称型であるとは、係数が

$$\alpha_i = \alpha_{p-i}, \quad \beta_i = \beta_{p-i} \tag{129}$$

を満たすということである。リープフロッグをこの形にしたものは、

$$x_{i+1} - 2x_i + x_{i-1} = h^2 f_i \tag{130}$$

というものであり、これが上の対称性の定義を満たしていることはいうまでもないであろう。

上の意味で対称な公式は、実際に時間反転に対して対称であるということに注意しておこう。つまり、$(x_i, ...x_{i+p-1})$ から上の式で x_{i+p} を求めたとする。時間を逆転させた解を考えて、$(x_{i+p}, ...x_{i+1})$ から x_i を求めるということを考えても、使う式が時間が逆転していないものと同じになっているのである。

このような公式が存在することは1970年代から知られていたが、注目されるようになったのはシンプレクティック法の研究が盛んになった1990年代に入ってからである。特にトロント大の Quinlan と Tremaine[QT90] が14次までの対称型公式を導いている。

なお、非常に最近になって、国立天文台の福島によって、Quinlan & Tremaine によるものよりもはるかに良い性質を持つものが導かれている。[Fuk98]

ただし、これらの公式に共通する欠陥として、線形解析では安定であっても、非線形問題（例えば、単純なケプラー問題）に適用すると共鳴により不安定な振動を起こすことがあるという困難があり、これはまだ未解決の問題である。[Fuk99, ET99] このために6次以上の公式は現在のところ実用になっていない。

福島はこれを解決する方法として、「超陰的方法」というものを提案している。理屈はある意味簡単で、x_{i+p} を求めるのにもっと未来の情報まで使うというものである。もちろん、このためには複数ステップを「一度に」解く必要があり、初期値問題をあたかも境界値問題であるかのように連立方程式として解くことになる。こういった方法が実用になるかどうかはまだこれからの研究の発展を見ないとわからない。

6.7.10 エルミート公式

さて、前に紹介したエルミート公式だが、一般の次数の場合にはそうではないが、広く使われている4次の公式は実は時間対称である。ちなみにシンプレクティックではない。といっても、PEC モードで使っていたのでは陰公式が厳密に満たされてないので、時間対称公式のもつ良い性質ははっきりとは現れない。極限修正モード、あるいは $\mathrm{P(EC)}^2$ や $\mathrm{P(EC)}^3$ モードで使うことで実質上時間対称性を実現することができる。

とはいえ、普通に独立時間刻みと合わせて使っていたのでは、シンプレクティックの場合と同様に時間刻みを変えることが時間対称性を破る。また他の粒子の軌道を予測する時には過去の情報しか使っていないので、ここでも対称性が破れている。このために、通常のエルミート公式と独立時間刻みを合わせたやりかたでは、$\mathrm{P(EC)}^2$ にしても特にいいことはない。

いいことがあるのは、可変時間ステップとはいってもほとんどの場合に時間刻みは一定であるような場合である。具体的には、惑星形成のシミュレーションでは、多数の微惑星が重力相互作用しながら太陽のまわりをほとんど円軌道で回っている。これらの時間刻みはほとんど一定であり、そのために P(EC)^{2}にするだけで劇的に計算精度が向上する。さらに、誤差のほとんどが太陽重力から来ているので、粒子間の相互作用は一度計算するだけで、太陽重力についてだけ反復することで精度をあげることができる。[KYM98]

6.7.11 時間刻みの対称化

さて、シンプレクティック公式は時間刻みの変更と相性が悪いのはすでに述べたとおりだが、対称型公式も普通にやったのでは同じである。が、対称型公式では「時間刻み対称化」によって良い性質を取り戻すことができることがわかっている。ここでは時間刻み対称化について簡単にまとめる。

可変刻みの場合、対称型公式であったとしても普通は「時間を反転して逆に積分すれば元に戻る」という性質を満たさなくなる。理由は簡単で、時間刻みが同じになるとは限らないからである。つまり、時間刻みは普通は前のステップでの誤差とかそういったものを使っているので、行きと帰りでは違ったものになる。

時間刻みが対称でないというのが永年誤差がでる原因だとすれば、対称にしてしまえば永年誤差がなくなるかも知れない。対称にするとは、時刻 t_0 で計算した時間刻み Δt_0 と逆に戻る時に t_1 で計算した時間刻み Δt_1 が等しくなるようにするということである。このためには、時間刻みが時刻 t_0 と時刻 t_1 の情報を同じ重みで使っていれば良い。一つの可能性は以下のようなものである。

$$\Delta t = \frac{1}{2}[h(\xi_0) + h(\xi_1)] \tag{131}$$

ここで ξ_0 は時刻 t_0 での位置と速度であり、$h(\xi)$ は適当な時間刻みを与える関数である。つまり、積分の始点と終点で時間刻みをそれぞれ計算して、その平均を取るわけである。こうすれば、逆に戻った時にも時間刻みが等しくなることが保証される。ここで注意しないといけないのは、ξ_1 は Δt の関数であるということである。このために、式 (131) は Δt を陰的に決める方程式になっていて、何らかの方法で解く必要がある。これには疑似ニュートン法などいろいろな方法が使える。

Cano と Sanz-Serna [CSS97] は、周期運動するハミルトン系について、(a) 時間刻み一定のシンプレクテック公式, (b) 時間ステップ対称化した対称型公式、(c) ハミルトニアンを保存するように構成された公式の 3 つについて、任意の量の誤差が最大でも時間の 1 次でしか増大しないということを証明した。これは時間刻み対称化にある程度の理論的裏付けを与えるものと解釈できよう。

自己重力系を応用先として考える時には、時間対称化が有効なのはまずは連星系等である。共鳴状態の 3 体のような、直接に数値積分する以外に特に有効な手段がなく、しかも時間刻みを変えることは絶対的に必要な場合によい結果を与えるからである。しかし、独立時間刻みと組み合わせることはできないだろうか？

Quinn et al. [QKSL97] は、リープフロッグは対称化した上で独立時間刻みを使うことが可能であることを示している。これは、基本的にはリープフロッグが陽公式であり、位置は予測子が修正されないでそのまま使われるためである。なお、速度は修正子によって修正される。従って、対称性を保証するためには、時間刻みが速度ではなく位置だけに依存するようになっている必要がある。彼らは時間刻みを $\sqrt{G\rho}$ に比例させることで速度に依存しない時間刻みを作っている。が、彼らの実験結果を見る限りでは、やはり密度だけによって時間刻みを決めることには無理があるのか、非常によい結果が得られているとはいい難い。そのせいかどうか知らないが論文もまだ通っていないようである。

6.8 ネイバースキーム

ここまでは時間積分の手法の話をしてきたが、ここでは加速度の計算の話になる。ツリーなどの近似算法を使ってある時刻での加速度を高速に計算するというのは次章の話題として、ここでは独立時間刻みをもう少し進めて、ある粒子への力をいくつかの成分にわけ、それぞれに違う時間刻みを与えるという方法について説明する。「いくつか」といっても、実用になっているのうは近くの粒子からの力と遠くの粒子からの力の 2 つに分けるもので、理論的にはそれ以上分けてもあまりメリットがないということがわかっている。[MH88]

分ける原理は、粒子毎に適当な半径 r を考えて、ある時刻でそのなかにいた粒子はネイバー粒子であるとし、リストをつくって憶えておくというものである。その時刻では、全粒子からの力をネイバーからの力 F_n と残りからの力 F_d に分けて計算しておく。タイムステップはこの 2 つのそれぞれについて計算する。普通は F_n のほうが短いステップサイズになるので、時間が F_d のタイムステップを超えるまでは F_n はステップごとに計算するが、F_d の方は外挿する。超える手前の時刻で、F_d を計算しなおして修正子を適用すると同時に、ネイバー粒子のリストも作り直す。

なお、実際にこのアルゴリズムを線形多段階法と組み合わせるのはかなり面倒である。というのは、F_d を新しく計算した時点でネイバー粒子のリストも変わるので、単純な差分によって予測子を構成できないからである。広く使われていた Aarseth によるプログラム [Aar85] では、入れ替わった分の補正を入れ替わった粒子について解析的に高次の導関数を計算して行なう。この処理はかなり繁雑なものになる。

但し、4 次のエルミート法の場合には予測子に前のステップの情報を使わないので、このような補正は不要になりプログラムを単純化できている。[MA92]

このやりかたでどの程度得になるかというと、理論的および数値的な解析の結果では、ネイバー数は $(N/10)^{3/4}$ 程度が最良で、計算量の比は $O(N^{1/4})$ 程度になる。[MH88, MA92] 粒子数の 1/4 乗というとたいしていいことがないように思うかもしれないが、これは必ずしもそうではない。というのは、コストにつく係数が小さいために比較的粒子数が小さいところ (100 程度) でも既に効果があるからである。粒子数が 10^4 程度では、計算コストに 10 倍程度の差が出てくる。

通常の計算機を使った高精度の重力多体計算では、現在現実的に可能な程度の粒子数ではこのネイバースキームを使った方法がもっとも高速であるようである。

6.9 ベクトル化、並列化

ここまでの話では、どういう計算機の上で実行するかということはあまり考えないで、精度を確保し、計算量（浮動小数点演算の数）を減らすにはどういうことを考えればいいかということを見てきた。30年前であればここまでで話はほとんどおしまいであり、あとはかなり細かいプログラミングテクニックの問題ということになった。

しかし、最近30年間の計算機の発展は、ある意味ものごとをより面倒にする方向に来ている。というのは、20年前であれば高速な計算機というのはベクトルプロセッサであり、アルゴリズムがベクトル化できるかどうかで性能が数十倍違ったし、現在ではもっとも高速な汎用計算機は分散メモリ型並列計算機になっていてここでは並列化できるかどうかで性能が千倍以上違ってくるからである。

独立時間刻みを使うような問題の並列化、特に分散メモリの計算機での並列化については、しかし、現在のところあまり研究がないのが現状である。これがどうしてかは良くわからないが、次に述べるツリー法に比べて本質的に難しいというのが一つの理由であろう。まず、計算量が多い割に扱う粒子数が小さく、効率的な並列化のためにはプロセッサ間の通信や同期などが非常に高速である必要がある。現在、並列計算機の主流になりつつある分散メモリ型並列計算機やPCクラスタがもっとも苦手とするタイプの計算なのである。これに対し、古典的なベクトル計算機や共有メモリ型の並列計算機ではあまり難しいことをしなくても性能がでる。

ベクトル計算機や共有メモリ型の並列計算機で単純な独立時間刻みを使う場合、力の計算には全粒子からの力を受けるという高い並列性があるので、そこのところを並列化すれば十分である。これに対してネイバースキームとなってくると並列度が下がるので、もう少し増やしたい。これにはブロックステップ法というものが使われる。[McM86] この方法では、基本的に時間刻みを2のベキ乗に量子化することで、なるべく沢山の粒子が同じ時刻をとるようにする。同じ時刻にくるものは並列に計算できる。

分散メモリの場合、今のところどのように並列化すべきかということがそもそも良くわかっていない。提案されているのは例えば全プロセッサが全粒子のコピーを持つ方法である。[SB99] 今のところ、この方法ではあまり高い性能が出ていない。使っているプラットホームはCray T3D/Eであり、プロセッサ間通信が非常に速いものであるので、それでも性能が出ていない理由はよくわからない。理論的には、この方法で通信が遅いPCクラスタで良い性能を出すのはきわめて難しい。

汎用並列計算機での効率的な実装の研究が進んでいないもう一つの理由は、最後に述べる専用計算機が単純な独立時間刻みに対しては極めてよい成果を出しているということかもしれない。これについては後でもう少し詳しく議論する。

7 計算法 — 空間領域

今までは、微分方程式の右辺は与えられている、つまり基本的に全粒子からの力を合計するとして話をしてきたが、ここでは右辺の全粒子からの力を近似を使って速く計算すると

いう話をする。

歴史的にも、また現在使われているものとしても、右辺を速く計算するための方法はたくさんある。が、基本的な考え方としては以下の3つにわけることができよう。

第一は、各粒子への力を直接計算するのではなく、空間格子を切って格子上でポアソン方程式を解き、それから補間などで各粒子への力を求める方法である。境界条件や粒子分布によってどのような格子をとるかは違ってくるが、3次元の問題では直方体領域で普通に等間隔格子を切ることが多い。これは、ポアソン方程式を解くのに高速フーリエ変換を使えるので、計算が速いからである。普通 PM (particle mesh) 法というときにはこの FFT を使う方法を指す。

この方法の利点は速度にあるが、欠点はフーリエ変換を使うために規則的な等間隔格子でないと使えないというところにある。このために空間分解能に制限がつき、格子間隔の数倍程度以下では粒子間重力は実効的にソフトになるし、また格子の方向による非等方性も出てくる。このために、例えば宇宙初期で構造が一様に近いといった、自己重力系としては限られた状況でしか使えない。

この分解能の問題を改善するアプローチももちろんある。昔からあるのは P^3M(particle-particle particle-mesh) 法というもので、近くの粒子からの寄与は直接計算しようというものである。この方法では、 $1/r$ ポテンシャルを2つに分ける。一つは、$r \to 0$ の極限では $1/r$ だが遠くでもっと速く落ちる近距離力的な振舞いをするもの、もう一つはこの近距離力を引いた残りである。で、前者は直接計算し、後者は FFT を使って計算する。

この方法は、プログラミングが比較的容易でありまた精度もあげられる。もっとも、精度をあげようとすると、遠距離力が格子の非等方性の影響を受けない程度に近距離力の作用範囲を大きくする必要があり、比較的急速に計算量が増える。また、格子が等間隔であるために粒子の密度分布が非一様であると計算量が非常に急激に増大する。

これに対し、いわゆる AMR (adaptive mesh refinment) を使うという話も最近はある。これは、密度が高いとかポテンシャルが急に変わるとかいうところでは格子を細かくしようというものである。AMR にしてしまうと普通には FFT が使えないので、 CG 法等の反復解法でポアソン方程式を解くことになる。

AMR と P^3M を組み合わせて近くの粒子からの力も正確に計算しようという話もあることはある。[CTP95] 但し、 AMR の場合にはメッシュを使って計算した力の $1/r$ ポテンシャルからのずれを正確に評価するのは困難になるので、理論的にはむつかしい問題がある。

第二の方法は、基本的には第一の方法と同じようにポアソン方程式を解くが、格子を切って差分法で解くのではなくなんらかの直交関数系に展開する。といっても、FFT でもそのような展開をしているわけで区別が明確とはいい難いが、ここでは実際に考えるのは球面調和関数展開である。

銀河とか星団とかは、おおむね丸い形をしているので、それを表すのに球面調和関数展開を使うのは考え方としては非常に自然である。とはいえ、実際にこの方法を使うに当たってはいろいろ問題がある。根本的な問題は計算量である。FFT のようなうまい性質がないので、普通にやると計算量は展開項数と粒子数の積に比例する。従って、角度方向や半径方向の分解能を粒子間隔程度にとると、結局展開項数は粒子数の程度になり計算量は粒子

数の 2 乗程度になる。従って、項数をあまり多くとることは現実的ではない。

実際的な問題としては、 r 方向の展開をどうするかということがある。r 方向の積分について直交する関数ならなんでもいいわけではあるが、現実問題としては上の項数の問題があるので、あまりなるべく少ない項数で系を表現したい。このために、もともとの初期条件の分布に近い式を 0 次にもってきて直交関数系を作るような怪しい方法がいくつか提案されている。[HO92] しかし、項数を制限するということは、系の可能な進化がその有限項で表されるもののみに制限されるということでもあり、こういうことをやった時に本当に追跡したい進化が表現できているかどうかは明らかではなくなる。

もう一つの方法は、半径方向は展開しないということである。つまり、ある半径での重力ポテンシャルを、内側からの寄与と外側からの寄与に分けると、それぞれは多重極展開で表現できる。内側からの寄与は中心に近い粒子から順番に計算できるし、外側からの寄与はもっとも外の粒子から順に計算できる。

計算量の観点からは、この方法は計算量が角度方向の項数と粒子数の積ですみ、しかも半径方向については展開による計算誤差が入らないので、3 方向に展開する方法よりも明らかに優れている。このために、1980 年代には単独の銀河の大粒子数計算にはよく使われた。[Vil82, AW86] 但し、並列計算を考えると、順番に計算しないといけないというのが嬉しくないので最近はあまり使われていないようである。といっても、プロセッサ数がよほど多くなければ並列化できないところはたいしたことはないので、使われなくなった本当の理由は良くわからない。この方法の問題は、単独で球に近い銀河でも、やはり AMR と同じくどのような誤差が入っているのか良くわからないということである。

と、いうことで、以下の話の本題であるツリー法系の計算法の話にはいる。ツリー法の基本的な考え方は、計算量を減らすために重力を及ぼす遠くの粒子を適当にまとめるということである。まとめるのに、階層的な木構造を使うことで、どの粒子に対しても小さなコストでうまいまとめ方を構成する。

さらに、重力を受けるほうも、近くにある粒子は同じような場を感じるわけなので、まとめて計算することが考えられる。重力を及ぼす方は多重極展開ができる。受けるほうも、同様に場を（球面）調和関数展開して計算する。こちらは通常局所展開と呼ばれる。

この両方をやるのが FMM であり、多重極展開はするけれども局所展開は使わないで多重極展開をそれぞれの粒子の位置で評価するのがツリー法である。

どちらの場合でも、粒子がある空間を分割することで多重極展開や球面調和関数展開を使えるようにする。同じ球面調和関数展開を使うにしても、系が「丸い」ということを使って一箇所で展開するのではなく、分割したものをそれぞれ展開するので系が丸くなくても使えるわけである。

7.1 ツリー法の考え

以下、具体的にどのようにツリー構造を構成し、それによってどのように相互作用を計算するのかを簡単にまとめる。

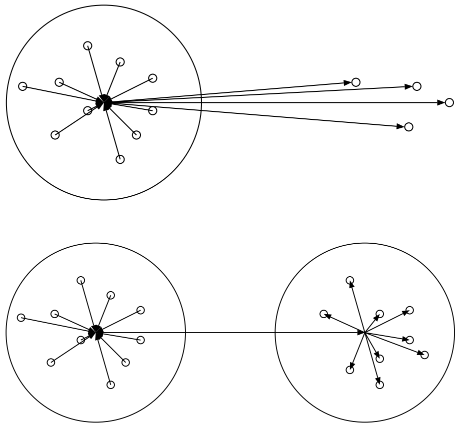

図 9: ツリー法と FMM の基本的考え。

図で説明しよう。3次元の図は書きにくいし見る方も見にくいであろうから、2次元で書くことにする(図10)。まず、平面内の適当な領域に粒子が分布しているとしよう。粒子が有限個なので、そのすべてを含む正方形を考えることが出来る。すべて含んでいればよいので、別に最小である必要はないが、普通はそこそこ小さくとる。これを、まず4つの小正方形(セル)にわける。で、そのそれぞれをさらに4つにわけるというのを再帰的に繰り返していく。

粒子が0個または1個しかなくなったところで止める。この空間(平面)分割に対応したツリー構造を考えると、根に全体の正方形に対応する節点があり、その下に4つの小正方形、さらにその下にそのそれぞれの中の4つ$\cdots$と続いていって、ツリーの葉には粒子1つ1つがくることになる。この方法では、粒子の数密度が高いところでは細かく、そうでないところでは粗いという意味で、適応的な構造が実現されていることに注意してほしい。3次元にすれば、正方形4個の代わりに立方体8個となるが、それ以外は全く同じである。

まず、どうやってこのツリー構造を使って一つの粒子への重力が計算できるかということを考えてみる。このためには、ツリーのある節点が表す立方体内の粒子全体からのある粒子への力を計算する方法を与えればよい。(なお、以下では、節点に対応する立方体、あるいはそのなかの粒子全体のことも単に節点ということがある)系全体からの力は、単に根節点からの力として計算できる。一つの節点からの力を以下のように計算することにする

$$\text{ある節点からある粒子への力} = \begin{cases} \text{その節点の重心からの力} \\ \text{(節点と粒子が「十分離れている」)} \\ \text{その節点の子節点からの力の合計} \\ \text{(それ以外)} \end{cases} \tag{132}$$

「十分離れている」かどうかの判定には通常は以下のような基準を使う。

$$\frac{l}{d} < \theta \tag{133}$$

ここで d は粒子と節点の重心の距離、l は節点に対応する立方体の一辺の長さである。パラメータθは見込み角といわれる量であり、計算精度と計算量を制御する。高い精度を得た

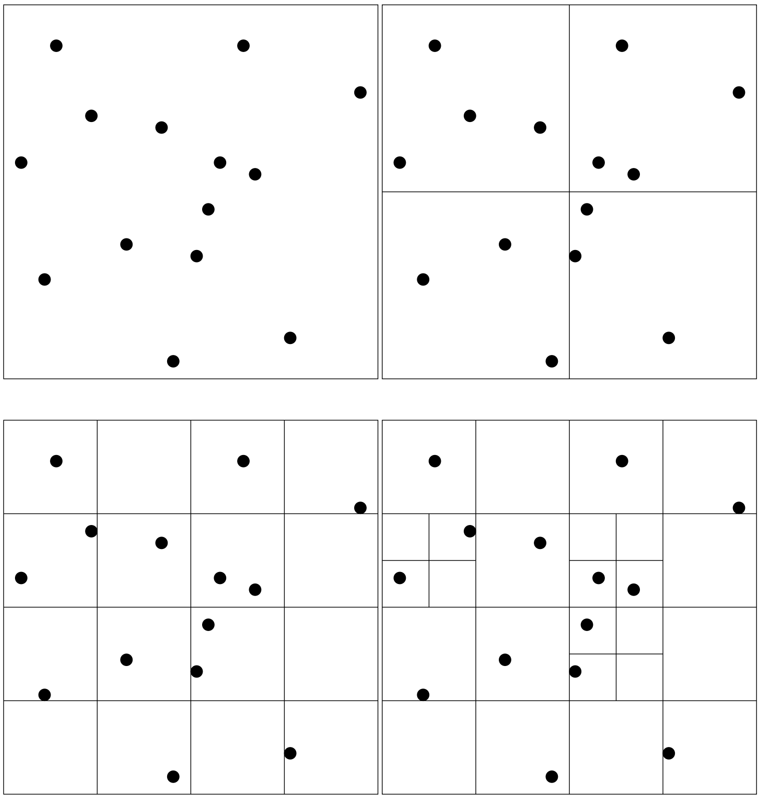

図 10: ４分木の構築

ければ θ を小さくすればいいが、漸近的には θ^{-3}に比例して計算量が増える。計算精度をあげるもう一つの方法は、重心だけを考えるのでなく多重極展開を作っておくことである。

原点中心で半径 a の球内にある粒子が球の外に作るポテンシャルの多重極展開係数は

$$\alpha_l^m = \sum_{i=1}^{N} m_i \left(\frac{r_i}{a}\right)^l Y_l^{-m}(\theta_i, \phi_i) \tag{134}$$

で与えられる。ここで、m_i と (r_i, θ_i, ϕ_i) はそれぞれは粒子 i の質量と極座標で表した位置である。また、$Y_l^m(\theta, \phi)$ は l 次の球面調和関数で、ルジャンドル陪関数 P_l^mを使って

$$Y_l^m = (-1)^m \sqrt{\frac{2l+1}{4\pi}\frac{(l-|m|)!}{(l+|m|)!}} P_l^m(\cos\theta) e^{im\phi} \tag{135}$$

と書ける。この展開係数を使うと、位置 (r, θ, ϕ) （$r > a$）でのポテンシャルは

$$\Phi(r, \theta, \phi) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \alpha_l^m \frac{a^l}{r^{l+1}} Y_l^m(\theta, \phi). \tag{136}$$

と書けることになる。

ツリー法を実際に使うには、各節点についてその中の粒子全体の重心の位置と質量（あるいは多重極展開の展開係数）をあらかじめ求めておく必要がある。これらはやはり再帰的に定義することができる。つまり、子節点の中の粒子全体の重心の位置と質量がわかっていれば、親節点の情報は計算できる。多重極展開の展開係数についても（計算は繁雑であるが原理的には）おなじことである。実際の計算プログラムにおいては、再帰手続きで実現してもよいが並列化などを考えると葉節点から根節点に向かって順に計算していくほうがよい。

7.2 FMMの原理

ツリー法の原理からFMMの原理にはほんの一歩である。ツリー法では、ある粒子への力を計算するのに、遠くの粒子からの力はまとめて計算することにした。相互作用は対称的であるので、逆にある粒子から遠くの粒子への力をまとめて計算するということも考えられる。両方を同時にやれば、ある一群の粒子から別の一群の粒子への力をまとめて計算することになる(図9)。このように、受ける側もまとめてやることによって、以下に説明するようにツリー法の $O(N \log N)$ から $O(N)$ へ計算量のオーダーを下げることができる。このためにこの方法には「高速」多重極展開法という名前がついている。

力を受けるほうは、重力ポテンシャルのテイラー展開を求めておいて、それを各粒子の位置で評価するという操作が必要になる。もちろん、実際にはポテンシャルは調和関数になっているので、球面調和関数で展開することで項数をテイラー展開に比べて減らすことができる。この展開のことを、以下局所展開と呼ぶ。

この方法は、Rokhlin と Greengard によってまず2次元の場合に提案され[GR87]、すぐに3次元に拡張された[GR88]。考え方は2次元でも3次元でも同じであるので、例によって

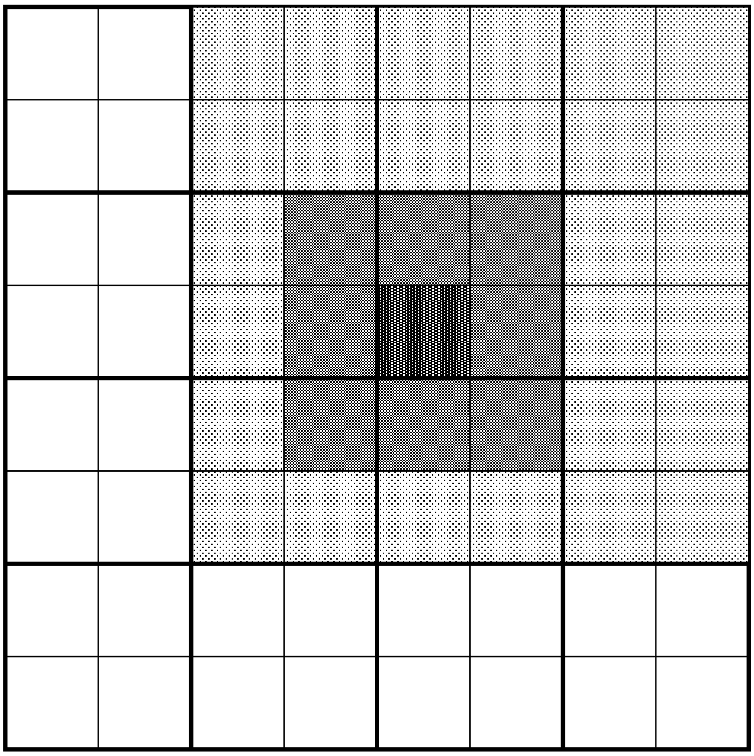

図 11: FMM の概念。もっとも濃く塗ったセルに入っている粒子への力を評価する。自分とその回りの 8 個のセルからの寄与は直接計算し、さらにその回りの 27 個は多重極展開を使う。その上のレベルは親セルが面倒を見る。

図は 2 次元で示す（図 11）。ツリー法では適応的なツリー構造を扱ったが、FMM ではとりあえず一様なツリー構造を考える。すなわち、ツリーのレベルを k とすると、元の正方形を 4^k 個の小正方形に分割する。なお、適応的なツリー構造の場合のアルゴリズムは例えば [BHER95] をみられたい。ツリー法の場合と同様に、あらかじめ各セルに含まれる粒子（一つも無いこともありえるが）が作るポテンシャルの多重極展開は準備しておく。

FMM では、一つの粒子への力を以下のように分割する

$$\boldsymbol{F} = \boldsymbol{F}_d + \boldsymbol{F}_c \tag{137}$$

ここで、$\boldsymbol{F}_d$ は直接計算する分で、自分のいるセルと、それに隣接したセルの粒子からの力である。これらには多重極展開は使えないので、直接計算することになる。残るのが $\boldsymbol{F}_c$ であるが、これはさらにツリーのレベル毎に別に計算する。つまり、あるセル i に対し、(i) そのセルと同じレベルにあって、(ii) その親セルとその隣接 8 セルの合計 9 セルの子であって、(iii) 問題のセルには隣接していない、$27\ (= 36 - 9)$ 個のセルを考える。この 27 個のセルに含まれる粒子が作る重力ポテンシャルの局所展開の和を、セル i の中心で評価しておく。トップレベル（系全体）と、そのすぐ下のレベルでは、親のレベルでは隣接していて自分のレベルでは隣接していないものは存在しないので、計算の必要はない。

このようにして各レベルで重力場を計算しておくと、あとは粒子の位置で、それぞれのレベルでの局所展開を評価して合計すればいい。が、別に粒子毎に全レベルでの展開を評価

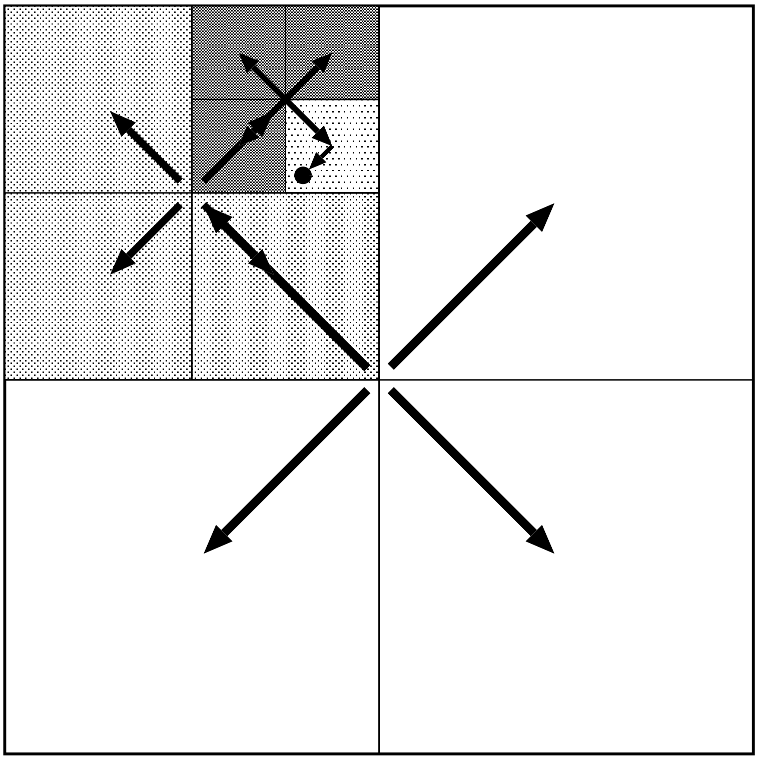

図 12: 局所展開のシフトと加算

しなくても、上のレベルの展開は一段下に移してやって（展開の中心をシフトして）まとめておけば、粒子は全部まとめた局所展開を一度だけ評価すればいい。この、局所展開をシフトしながら下に落していくのは、最初に多重極展開をやはりシフトしながら上にあげていくのとちょうど対称的な操作になる。

図 12 のように、4 レベルのセルの中心での展開を合計する場合を考えてみる。まず、一番上のレベルのセルの中心での局所展開を、4 つの子セルの中心に展開中心をシフトしてそれぞれのレベルでの重力場の局所展開と足し合わせる。さらに、それぞれの子セルで、合計した局所展開をまたそれらの 4 つの子セルの中心にシフトし、またそこでの局所展開と足し合わせる。これを最下層のセルに達するまで繰り返すと、各セルの中心での、自分と自分に隣接するセルに入っている粒子を除いた他のすべての粒子による力の局所展開がもとまる。粒子一つへの力は、この最下層での局所展開を粒子の位置で評価したものと、隣接セルの粒子からの力の合計ということになる。

ここで計算量のオーダーを考えてみると、すべての操作の計算量が粒子数またはセルの数にしか比例していないことがわかる。セル数は粒子数に比例するので、結局計算量のオーダーが $O(N)$ ということになるわけである。

7.3 それぞれの方法の歴史と現状

FMM とツリー法のどちらも、まだ10年少々の歴史しかない新しい方法である。しかし、どちらも急速に研究が進み、応用分野も粒子系のほか、境界要素法、あるいは変わったところでは非常に高次の球面調和関数展開や、分散メモリの計算機上での FFT など、さまざまな分野に広まりつつある。基本的な考え方は、遠くとの相互作用はまとめるという単純なものなので、重力・クーロン相互作用に限らず非常にさまざまな応用がありえるわけである。基本的には、相互作用が重ね合わせ可能でかつ遠距離に届き、といっても遠くにいくに従って弱くなるようなものであればなんでも FMM やツリーの考え方は使える。

ここでは、特に筆者の専門である重力多体系の観点から過去の歴史を簡単に振り返ってみたい。

ツリー法を初めて実現したのは、プリンストン大学の学部生 A. Appel であった [App85]。これは1981年の学部の卒業論文であったということだが、1985年になるまで公表論文にはならなかった。彼の方法は、ツリーを最近接粒子をまとめていくことによってボトムアップに構成していくものであり、現在広く使われている8分木をつかうものではなかった。8分木を使うツリー法を提案したのは当時プリンストン高等研究所にいた J. E. Barnes と P. Hut である [BH86]。この方法に基づいて、その後の数年間にベクトル型スーパーコンピュータへの効率的な実装についての研究と分散メモリ型計算機への実装の研究が精力的に進められ、銀河や銀河団、あるいは宇宙の大規模構造などの形成、進化のシミュレーションに広く使われるようになった [WS92]。特に分散メモリの超並列計算機では、10^9 を越えるような粒子数のシミュレーションが可能になってきている。

このように広く使われることになった理由の一つは、実装が比較的簡単であったことである。ツリー法の実装のほとんどは、4重極モーメントまでしか扱わない。このため、多重極展開や球面調和関数といってもそれほど面倒な計算式が出てくるわけではない。高い計算精度が必要であれば、離れたセルもある程度まで分割して相互作用を計算する。もちろん、非常に高い計算精度が必要になれば、より高次のモーメントをとり入れた方が計算量が減る。しかし、多くの問題で、粒子数が有限であることからくる2体緩和で計算精度がきまっているので、相互作用の計算精度をあげるよりも、計算精度を下げても粒子数を増やしたほうがよりよい結果が得られるのである。このために、かなり広い応用で、4重極モーメントよりも精度をあげることには意味がない。これについては後でもう少し詳しく議論する。

このように、理論的にも実験的にも複雑なことをして精度をあげる必要がなかったために、ベクトル化や並列化、特に適応的な木構造の実現やロードバランシングなどに努力が傾注されてきた。これらの努力により高速で実用的なプログラムが利用可能になってきている。

FMM はツリー法とほぼ同時期にイェール大学の Greengard と Rokhlin により提案された [GR87, GR88]。これは多重極展開と局所展開をまともに使うものであった。その後、計算法の改良について非常に多様な提案がなされてきた。例えば多重極展開から局所展開への変換については、FFT を使って高速化する方法 [EJ96] や、一旦座標系を回転し、シフトの方向に z 軸をもってくることで計算量を $O(p^3)$ に減らす方法 [WHG96] などが提案されている。この他にも最近 Greengard らによって提案された新しい方法 [GR97] もある。これ

らを使うことで、理論的な計算コストは $O(Np^2)$ よりも若干小さくなっている。しかし、このような多様な提案があるということは、逆にいえばまだ決定版というべきものはなく、多数の改善案が提案されているがそれらのうち一つが他のものよりも圧倒的によいというわけではないということでもある。さらに、このような方法は一般に高い精度を要求する場合にしか改善につながらない。

実際のところは、天文シミュレーションにおいては現在のところ FMM は全く使われていない。このように、理論上の計算量はともかく、現実の問題としては FMM があまり使われていない一つの理由は FMM の実装が面倒であるということである。ツリーでは木構造を作った後は多重極モーメント (といってもほとんどの場合せいぜい 4 重極) を計算し、あとはそれを使って各粒子への力を計算する再帰的な関数を書けばおしまいだが、FMM の場合には多重極展開を局所展開に変換するややこしい式と、局所展開をシフトする式をプログラムしないといけない。しかも、特に多重極展開を局所展開に変換するところはまともにやったのでは計算量が多いし、といって減らそうと思うとまたいろいろ面倒なことをしないといけない。

もちろん、それだけやって実際に同じ計算精度で速くなるとか同じ計算時間でより精度がよくなるとかいうことがあれば、自分で書かないまでも誰かが作ったプログラムを使おうかという気にもなるが、現在のところ実際に精度と計算速度を比較した研究では、少なくとも比較的計算精度が低いところではツリーの方が速いという結果が得られている。例えば、適応的な木構造の場合にツリー法と FMM の計算速度を比較した研究 [BS97] では、比較的精度が低い (ポテンシャルの相対精度が 10^{-3}程度) 場合、FMM はツリー法に比べて有意に遅いという結果になっている。となると難しいことをわざわざしないほうが得であろう。

もっとも、現在のところ、ツリー法に比べて FMM の結果があまりよくないのは少なくとも部分的には FMM の実装があまりよくないからであると理論的には思われる。例えば、前節の説明では、多重極展開を局所展開に変換するところであるセルは自分と同じ大きさのセルからの力を受ける。で、どちらも同じ次数で展開する。誤差の上限の評価ではこれは同じオーダーで係数も同程度だが、実際に入ってくる誤差を考えると必ずしもそうはならない。というのは、多重極展開のほうは、あるセルに入っている沢山の粒子からの力の合計であるので、高次の項は主にセル内の密度の揺らぎから来ている。このために、中の粒子数が大きければ揺らぎが小さくなり、その結果誤差も小さくなる。ところが、局所展開ではあくまでもセル内の一箇所での評価なので、展開の打ちきり誤差がそのまま入ってくる。中に沢山粒子があっても、それぞれに独立に誤差がでるだけである。このために、多重極展開と局所展開で実際の誤差に対する寄与が同程度であるためには、局所展開のほうの次数をかなり高くするか、あるいは局所展開のほうのセルを小さくする必要があると考えられる。ところが、現在のところ大抵の実装ではこういった効果まで考慮して計算量を最適化しようとはしていない。

なお、2 次元の場合には 3 次元に比べて FMM の計算量は大きく減り、実用性の高い方法となる。これは、多重極展開の項数が p^2 から p に減るだけでなく、幾何学の違いのためにセル間相互作用の数が減るためである。3 次元の場合は一つのセルが $6^3 - 3^3 = 189$ 個のセルと相互作用するが、2 次元の場合は $6^2 - 3^2 = 27$ 個であり、7 倍計算量が違うことにな

る。なお、もちろんツリー法でも同じ理屈で減る。

幾何学の違いは、空間が 3(2) 次元であっても実際に計算する領域は 2(1) 次元である境界要素法の場合にも有利に働くことを注意しておきたい。このために、FMM やツリー法は境界要素法には特に有効である。

7.4 FMM の効率的実装

FMM の計算コストを減少させる方法についてはいくつか簡単に述べたが、ここでは実装を簡単にする (必ずしも計算量は減らない) 方法を 2 つ紹介しておく。これらは、プログラムが簡単に実現できるという意味で、FMM が広く使われるためには重要なものと考えられる。

7.4.1 アンダーソンの方法

Anderson [And92] は球面調和関数を使わないで FMM を実現する画期的に安直な方法を提案した。彼のアイディアは、球面調和関数の展開係数をデータとして使う代わりに、球面でのポテンシャルの値そのものを使おうというものである。半径 a の球の中の粒子が外に及ぼすポテンシャルは、球の表面でのポテンシャルの値がわかっていればラプラス方程式の境界値問題を解けば与えられる。解はポアソンの公式を使って以下のように書ける

$$\Psi(\boldsymbol{x}) = \int_S \sum_{n=0}^{\infty} \frac{2n+1}{4\pi} \left(\frac{a}{r}\right)^{n+1} \mathrm{P}_n \left(\frac{\boldsymbol{s}\cdot\boldsymbol{x}}{|x|}\right) \Psi(a\boldsymbol{s}) ds, \tag{138}$$

ここで P_n は n 次ルジャンドル関数である。この積分を、球面上での適当な標本点を使った数値積分に置き換える。2 次元であれば、円を等分割すればいいが、3 次元ではうまく点をとってその上で球面調和関数の直交性が保存されるようにする必要がある。これは良く研究されていて、文献 [McL63, HS96] を見れば数値的にそのような点が求められているので、それを使えばよい。

子セルの外の多重極展開をその一つ上のセルでのそれに変換するには、上の式に従って親セルを含む球面上でのポテンシャルを求めればいい。また、局所展開も、$1/r$ での展開が r での展開に変わるだけで上と同じ式である。さらに、多重極展開を局所展開に変換するには、上の式をつかって局所展開する球面上でポテンシャルを評価すればよい。つまり、上の式だけで、FMM に必要な数学がすべて表現されていることになる。

7.4.2 $\mathrm{P}^2\mathrm{M}^2$法

ここで、著者らが最近提案した $\mathrm{P}^2\mathrm{M}^2$法 (pseudo-particle multipole method)[Mak99, KM01] にも簡単に触れさせていただきたい。基本的な考え方はアンダーソンの方法とほとんど同じものであるが、違いは球面上でのポテンシャル分布を使うのではなく球面上の質量分布によって多重極展開を表現するというものである。定式化はアンダーソンの方法とほとん

ど同じであり、最終的には

$$m_j = \sum_{i=1}^{N} m_i \sum_{l=0}^{p} \frac{2l+1}{K} \left(\frac{r_i}{r}\right)^l \mathrm{P}_l(\cos\gamma) \qquad (139)$$

という形で実粒子の分布から仮想粒子の質量が表現される。ここで m_i, m_j はそれぞれ実粒子、仮想粒子の質量であり、γ は展開の中心から見て実粒子と仮想粒子がなす角度、r_i は実粒子の原点からの距離、r は仮想粒子を置く球の半径、K は仮想粒子の数である。

この方法は、FMM よりもむしろツリー法との組み合わせで有効であり、粒子間相互作用を計算するだけで高次精度のツリー法が実現できる。

もっとも、この方法は計算量という面ではあまり有利とはいい難い。これは、最小限必要な疑似粒子の数に比べて実際に使う疑似粒子数がかなり多くなるからである。例えば、もっとも低次の重心（1 次のモーメント）だけを使う場合を考える。疑似粒子とかそういうことを考えなくても、重心に一つ粒子をおけばいいというのは自明である。ところが、球面調和関数展開してから 1 次の項をというようなことを考えると、4 個必要になることになる。これが 4 重極になると、必要な粒子数は 12 に増える。通常の計算法では、4 重極モーメントを計算するのに必要な計算量は粒子一つからの力を計算する場合のせいぜい 2 倍程度なので、12 というのは普通に考えると現実的な計算量とはいい難い。

必要な疑似粒子数が多いのは、粒子の空間配置を固定しているからである。空間配置を固定して、質量だけを変えることで、線形計算で疑似粒子を決められるというメリットはあるものの、粒子あたり 4 つの自由度のうち一つしか使っていないので、最大 4 倍の粒子が必要になってしまう。

1 次の場合には上に述べたように「重心に粒子を置く」というのでいいわけだが、2 次以上に場合にもなんとかならないかというのはそういうわけで検討に値するであろう。今のところ、2 次 (4 重極) の場合にはそういうものができるということがわかっている。

2 次のモーメントまで考えた時には、独立な係数は 9 個である。従って、粒子 2 個では不可能だが 3 個ならできそうな気がする。3 個の粒子の質量の合計はもちろん実粒子のそれに等しい必要があるし、重心も一致しなければいけない。これで自由度は 4 個使うので、残っているのは 8 個である。この 8 個で四重極モーメントテンソルの独立成分 5 個を表せればいい。

4 重極モーメントテンソルが対角化される方向に座標系をとると、座標系で 3 自由度使ったので残りは第一軸方向のモーメントと第二軸方向のモーメントの 2 つだけである。例えば 3 粒子の質量を等しくし、それを第一軸か第二軸のどちらかに底辺が平行な二等辺三角形配置にすることで、対角化の条件は満たされるのであとは辺の長さをモーメントの値が等しくなるように決めればいいことになる。もっと違う配置ももちろんあり得るが、とりあえず四重極モーメントを合わせるにはこれで十分といえる。

この方法は実装ができていて、速度はもちろん 12 個も粒子を使うのに比べれば断然速い。

4 重極（2 次）が理論上可能な最小の粒子数で表現できるなら 8 重極（3 次）も、、、というのが人情だが、今のところそういう方法は発明されていない。これは不可能とかそういうことではなく、単にまだ誰もそんなことが可能かどうか真面目に考えたことがないというだけである。

7.5 計算精度等

ツリー法の計算精度に関しては、良くわかっていない問題がいくつかある。ここで問題なのは、例によって「計算された重力に対してある要求した精度を実現するためにはどうすればよいか」ではなく、「どのような精度になっていればシミュレーション結果は正しいといえるのか」ということである。

例えば、式 (133) を使った時に、粒子分布とθ を与えれば求まった重力の誤差とかその分布は実験的にも理論的にもわかる。が、ここで問題なのはではどのような精度が必要かということである。もうちょっというと、それ以前にそもそも重力の誤差というのがシミュレーションの「正確さ」を測るのに適当な物差しか?という問題が、あるわけである。

ある粒子の軌道で、ある時刻 t において加速度 $F(t)$ を計算してある誤差 $e(t)$ があったとしよう。今、時間積分は無限に正確で誤差がないとし、また数値積分した結果の軌道のずれは十分小さくて、誤差の非線形性は無視できるとする。この時、例えばある時間経過した後の速度の誤差はもちろん $e(t)$ を積分したものであり、位置の誤差は速度の誤差を積分したものである。

位置や速度自体を正確に計算することにどれほど意味があるかというのは前に長々と議論したが、まあ例えばエネルギーの誤差とかいうことにしても話は同じで数値計算法や要求する計算精度によって変わるのは $e(t)$ であり、さまざまな量の誤差は $e(t)$ の時間積分である。

さて、こんなことはあまりにも当たり前なことではあるが、上で見たように誤差を決めるのは $e(t)$ の積分であり $e(t)$ そのものではない。従って、計算精度を問題にするなら $e(t)$ のある瞬間での値とかその 2 乗平均値ではなく、$e(t)$ の自己相関関数なりパワースペクトルをみないといけないわけである。仮に $e(t)$ の瞬間瞬間での値は大きかったとしても、それが短い時間スケールでランダムに変われば時間積分した時の効果は小さい。逆に瞬間瞬間での値は小さくても、長いタイムスケールでしか変わらなければ時間積分した時の効果は大きいかもしれない。従って、ある瞬間での $e(t)$ の値自体に意味があるとは限らない。

ここで問題なのは、それでは一体どのような尺度を使うべきかということであるが、これは問題に依存することは明らかであろう。

例えば、ビリアル平衡にある恒星系で、速度分散によって構造が維持されているものを考える。この時にツリー法で力を計算すると、ある粒子からの誤差の相関時間は、その粒子が相互作用計算の時に使われるセルを横断する時間の程度になる。もちろん、球面調和関数の次数が大きければそれに応じて相関時間は短くなるが、オーダーとしてはそうなっている。従って、2 粒子間の相互作用の誤差の相関時間は距離にほぼ比例するものと考えられる。というのは、2 粒子間の相対速度は距離に無関係だからである。

あ、ここで暗黙の仮定として、パラメータ θ が一定であるというのを使っているが、いろいろ最適化をはかった場合でもθの距離依存性は非常に小さいのでこれはまあ妥当な仮定である。

さて、相関時間が距離に比例するとすると、誤差自体への寄与は距離が近いものほど大きくなっても構わないことになる。もちろん、ここにはさらに仮定が必要で、相関時間が実際

に有限であるためには誤差の平均値が 0 でないといけないわけだが、これは一応成り立っていると仮定することにする。

さらに、かなり大胆な仮定であるがある粒子への他の 2 つの粒子からの力の誤差は無相関であるとする。これは粒子数が大きい極限では明らかに正しくないが、現在可能な計算で実現されている粒子数ではこう仮定して実際上問題はないようである。

この時には、各粒子からの力の誤差の時間積分の合計をとると、1 次の項は 0 になり 2 次の項は結局 2 体緩和によるエネルギー変化に 2 粒子間の力の相対誤差程度の大きさの係数がついたものが出てくることになる。[MIE90, HHM93] 従って、粒子の軌道の誤差は 2 体緩和によるものよりもずっと小さい。さらに重要なことは、これは結局 2 体緩和と同じ距離依存性を持つということなので、近くと遠くの計算誤差が同程度になっているという、計算量と計算精度の面から見ると最適な状況が実現されているということである。つまり、粒子分布が一様に近く、粒子のランダム速度が大きいという状況に関する限り、単純な幾何学的条件が最善であるということになる。

ここで、単純な幾何学的条件は賢くなさそうだとか考えて、ある瞬間での力の誤差を最小にするようなやり方を考えたとしよう。上の議論から明らかなように、幾何学的条件を使った場合にはある瞬間での力の誤差に寄与するのは主に近傍の粒子である。このために、そのような「改善された」やり方では必ず幾何学的条件に比べて近くをより正確に、遠くをよりいい加減に計算することで同じ計算量で精度をあげようとすることになる。これで確かに力の誤差は減るが、時間積分した軌道では遠くをいい加減に計算しているのが効いてかえって悪くなってしまうのである。

もちろん、現実の状況はもっと複雑である。例えば、2 つの粒子からの力が誤差が無相関であるという仮定は空間構造が一様でなければ明らかに正しくない。一般には小さいスケールでは一様とみなせても大きいスケールではそうではないわけなので、遠いからといって大きなセルにするのはあまりよろしくないかもしれない。

といっても、多くの銀河のように中心にいくほど密度が高く、遠くまで距離の冪乗で密度が下がるものを扱う時には、中心から非常に遠いところではそもそもほとんど質量がないので、相関時間とかいうこと以前に力自体が小さい。こういった状況では幾何学的条件は明らかに無駄といえる。

別の例を考える。よく行なわれる宇宙論的な構造形成計算では、宇宙初期には密度は一様、ランダム速度はほとんどゼロであり、誤差の相関時間は構造形成が進んで宇宙膨張から切り離された構造ができるまでの間ということになる。この場合には、相関時間が距離によらないので瞬間での力の誤差を最小にするような計算法がもっともよい。ところが、同じ宇宙論的計算でも、構造が発達して銀河や銀河団が出来てくると、時間積分した誤差を最小にするためには瞬間での力の誤差を最小にするような方法ではいけないことになる。

というようなわけで、現在のところ一つの条件でどんな場合でも使えるといった万能な計算精度の決め方はない。結局のところ、計算精度を変えてみて見たい物理現象に影響がなければ多分大丈夫であろうというような実用的な基準でチェックする必要がある。

7.6 並列化

重力多体系のシミュレーションにおいては、より大きな粒子数を使った計算をするということが極めて重要である。これは、多くの場合に計算精度が 2 体緩和で決まっていて、計算精度を改善するには粒子数を増やす以外に効果的な方法がほとんどないことによる。例えば銀河形成のシミュレーションで 2 体緩和によって出来た銀河の構造が変わってしまってはなにを計算しているかわからないわけだが、銀河の中心付近の高密度な領域、あるいは銀河円盤のように恒星がおもに回転運動していてランダム速度が小さいような領域では、現在数値計算で扱える粒子数ではまだ緩和時間が短かく、物理的に意味がある結果を得ることは困難である。このような領域では、物理的に意味がある結果を得るために第一にするべきことはなんらかの方法で扱える粒子数を増やすということになる。

扱える粒子数は、計算機の速度と計算法の効率で決まるので、計算法の効率をあげるというのと少なくとも同程度には計算機の速度をあげるというのも重要である。もっとも、後で簡単に触れるような自分で計算機を作るというようなある意味極端なアプローチをとらない限り、計算機は大学なり研究所の計算センターの計算機を時間単位で利用するか、あるいは自分の研究室のワークステーションやパソコンを使うという話になってそれ以上あまり考える余地がないような気もする。が、これは必ずしもそうでもなくて、実際には使える計算機に合わせて計算法自体を変更する、あるいは新しいものを開発するといった必要が出てくる。

特に、ここ 10 年ほどの高速計算機の発展の方向を見てみると、高いピーク性能をもつ計算機が安価に供給されるようになったことは事実であるがその上で利用可能なプログラミング環境はほとんど進歩しておらず、というよりは退化しており、シミュレーションプログラムで高い性能を得るための手間が加速度的に増大しているのが現状である。

特に話を面倒にしているのが、「分散メモリ型並列計算機の上でのメッセージ・パッシングプログラミングモデル」というものである。これは、要するに、並列計算機といってもそれぞれ独立なメモリと CPU をもったたくさんの計算機の集まりであり、それらが互いに通信できるというだけのものである。これはもちろん並列計算機では非常に古くから使われてきたプログラミングモデル、というよりは「計算機を作る側ではなにも考えないから、使うほうでなんか考えてよ」という、計算機工学の側の仕事放棄を表しているだけともいえる。

もちろん、そのようなモデル、というよりはモデルの不在が特に科学技術計算の分野で主流になったのは、使う側がそれを選択したからという面がある。問題が単純でこのようなものでもなんとかプログラムが書けてしまうような場合には、このようなやり方で高い性能が得られる。また、各プロセッサでは普通の Fortran や C で書いたプログラムが動いているだけなので、プログラムを書き始めるのに対する敷居はそんなに高くない。さらに、チューニングに関しても、基本的には従来の単一プロセッサの場合と同じなので特別なツールとかがいるわけではない。これに対してもっと高度な並列処理言語のようなものを使った場合には、多くの場合にプログラムの実行効率が下がるし、また高水準のプログラムが実際にどのようにハードウェアにマッピングされているかが隠されてしまうためにチューニングが容易ではなくなる。

とはいえ、ツリー法のように原理的にはすべてのステップが並列実行できる比較的素性のいいアルゴリズムでも、これを分散メモリ型の並列計算機に載せようとするとアルゴリズムのかなり根本からの変更が必要になる。以下、そのあたりを少し議論しよう。

ツリー法や FMM を分散メモリ型計算機に実装するうえで問題になるのは、空間分割をどのようにやるかということである。ツリー法は基本的に遠くのほうは適当に計算するという方法なので、物理的に近くにある粒子を同じプロセッサに割り当てることでプロセッサ間でのデータ交換を少なくすることができる。

ここで、粒子の分布が空間的にほぼ一様であれば、基本的に計算領域を規則的に同じ大きさに分ければ粒子数は均等になるし、またその結果計算量もおおむね均等に分配されて高い実行効率が期待できる。が、広い領域に銀河が1つとか2つだけあって、まわりにはほとんどなにもないという状況ではそうはいかない。

現在までに作られたほとんどのプログラムで使われているのが、再帰的直交分割 orthogonal recursive bisection という方法である。[WS92] この方法の原理は以下のようなものである。まず粒子をある座標軸方向に並べて真中で分ける。分けたものそれぞれについて、今分けたのとは直交する座標に並べてまた真中でわける、、、というのを、プロセッサ数になるまで繰り返す。実際に並べなくても真中あたりで分ければいいわけなので、実装はいろいろである。

こうして出来た分割を各プロセッサに割り当て、基本的には各プロセッサで独立にツリー構造を作る。

各プロセッサ毎にツリーが出来れば、自分のなかでの相互作用は普通と同じように計算できる。別のプロセッサにあるツリーからの力については、まず、「そこからの力を計算するのに必要になりえる範囲のツリー構造」を送ってもらって、そこからの力を普通の方法と同じように計算すればいい。このツリー構造は、基本的にはツリーの枝のあちこちを切り落としたものになる。つまり、ある節点のところで、すでにツリーを受けとる側のどこからみても十分に遠いという条件を満たしていれば、そこから下は決して計算に使われないので送る必要がなくなるのである。

もっとも、この方法では若干の無駄がでる。というのは、本来ならば複数プロセッサの粒子をまとめて1節点として重力が計算出来るような場合でも、そのような大きな構造を作っていないのである節点からの力は少なくとも1度は計算されることになるからである。また、ORB の境界はツリー自体のノード境界と無関係なので、本来同じ節点に属する粒子が2つのプロセッサにまたがる結果計算量が増えるという効果もある。

これらがどれくらい大きな損失であるかに関するきちんとした定量的な評価は残念ながら文献にはないが、プロセッサ数が非常に大きくなると問題になるのは明らかである。例えば、プロセッサ数が数百程度になると、プロセッサ毎に一つというのでも本来のツリーで必要な計算量よりも大きくなる。また、通信についても、データ量はともかく回数が増えるためのオーバーヘッドが無視できなくなってくる。

この問題を回避する方法は2種類知られている。一つは通常の8分木をやめて、各プロセッサの中で ORB をさらに繰り返し適用することでツリーを作るという方法である。8分木に比べてツリーを作るコストはかなり増えるが、まあ通常の実装ではもともとそこの計算

コストは小さいので大した問題ではない。

もう一つは、8 分木を使って、なんらかの方法で全体ツリーを作り直すことである。原理的にはベストな方法は、各プロセッサが他の全プロセッサから必要な情報をもらってきた後で、その情報を使ってそのプロセッサの粒子からみた全体ツリーを作るというものである。このためには、単に他のプロセッサから渡ってきたツリーの葉になっているものを全部粒子だと思って、もともと自分が持っている粒子とまとめてツリーを作り直せばいいことになる。

まあ、これは少なくとも自分のが既に持っている分については 2 度手間になる。普通は既にいったようにここの部分の計算時間は知れているので問題ではないが、ここの部分が問題になるようであれば減らすことは出来る。それには、ツリーを作る方法を変えればいい。

基本的なツリー法のアルゴリズムを説明したところでは、ツリーを作る方法として最初に全粒子が入る大きな立方体を考えて、それを分割していくという方法を紹介した。これはまあわかりやすい方法であると著者は思うが、 Barnes と Hut の最初の論文では全く違う方法が説明されている。先に説明した方法は、実は著者がベクトル化を可能にするために導入したもの [Mak90] である。

Barnes と Hut の方法では、まずからっぽの箱を作り、そこに一つずつ粒子を入れていく。ある箱に入っている粒子が 2 つになったら、その箱を分割して粒子を振り分ける。分割してもまだ同じ箱に入る場合ももちろんあり、その時は単に分割を繰り返す。

このアルゴリズムが実装されていれば、他のプロセッサから渡ってきた情報をまとめるには、それらの「粒子」を既にあるツリーに入れていくだけである。

余談になるが、ツリー法を共有メモリ型の計算機で並列化するという研究はかなりたくさんある。これらの多くは、計算機科学（並列処理）の研究者によってなされたものであった。その多くは ツリーを作るのに Barnes と Hut のもともとの方法を使って、それを非常な苦心をして並列化している。代表例は良く知られた Stanford 大学の SPLASH ベンチマーク [WOT+95] である。並列化するには複数の粒子を同時にツリーにいれたいわけだが、一つの粒子のために構造を変えるのと他のどれかのために構造を変えるのが干渉すると結果が変になってしまうので、同期とかロックとか難しいテクニックを使って矛盾が起きないようにする。もちろん、その結果として速度は低下し、ある程度以上プロセッサが増えても速度は上がらなくなる。これに対し、著者の方法は、ベクトル化も共有メモリ型の計算機での並列化もトリビアルであり効率も良い。なにがいいたいかというと、並列処理の研究者の場合、どうしてもアルゴリズムを所与のものとしてそれと「同じ」計算手順を並列に実行しようという方向に目が向く。このために、結果が同じでもっと簡単に並列化でき、さらに並列実行でない時でも性能差が問題にするほどはないような簡単な方法が作れる場合でも、そういったものを作るということにそもそも考えがいかないことが多いようである。

思いきり話がそれたが、このようにして全体のツリー構造を各プロセッサで作ってしまえば、そこから先の各粒子への重力の計算は各プロセッサで全く独立に行なうことができる。このために、ツリー法の並列化の効率はきわめて高く、プロセッサ数が 1000 を超えるような場合でも台数に比例した性能向上を実現した例がいくつかある。

7.7 独立時間刻との関係

さて、今までは、ツリー法の場合、時間刻みは全粒子同じで一定である、つまり、独立時間刻みは使わないという前提で話をしてきた。銀河全体のシミュレーションや、あるいは銀河団、宇宙論的構造形成の計算のように、一定の時間刻みとポテンシャルソフトニングを組み合わせるのが効率的な応用がいろいろあるのは確かである。しかし、球状星団のシミュレーションのようにそれではすまない応用もまたたくさんある。ここでは、そういった応用向けに、ツリー法と独立時間刻みを組み合わせる方法について検討する。

といっても、並列化、特に分散メモリの計算機での並列化を考えなければ、これはそれほど難しい話ではない。基本的には、ある粒子への重力を計算するのに、他の粒子すべてからの力を合計するわけではなくツリーをたどって計算するというだけなので、ツリーの節点に対しても予測子を持たせればいいわけである。重心運動については子節点（あるいは粒子）の予測子から容易に作れるし、4 重極も式は繁雑だが計算するだけである。ある粒子を動かした時には、その度にその粒子の上の節点の予測子はすべて更新する必要がある。とはいっても、手を抜くことはできなくはなくて、実際に予測子が呼ばれるまで計算しなおさないというふうにすれば多少速くなる。

もう一つ重要なのは、木構造が変わる時にどうするかということである。粒子が動き、自分のもともとはいっていたセルから出ていくので、それをほおっておくとセルの幾何学的大きさが粒子分布を反映しなくなる。このために、幾何学的な判定条件を使って重力計算すると誤差が大きくなってしまう。実際の粒子の広がりを計算しなおして、そっちを使って判定することで計算誤差の増加は防げるが、もともとのセルサイズよりも粒子の広がりは大きくなっていくので計算量が増える。従って、適当なところでツリーを作り直す必要がある。

もっとも簡単な方法は、適当な間隔で木構造をゼロから作り直すことである。宇宙論的シミュレーションや銀河形成のための計算コードでツリー法と独立時間刻みを組み合わせたものではほとんどこの方法がとられている。

これに対し、ツリー構造が変わったところだけ組み直すというのは実はそれほど難しくはない。これにもいくつかの方法があるが、一つは、ある粒子があるセルから別のセルに動いた時に、その 2 つのセルの共通の祖先であるセルの下は全部作り直すという方法である。これは McMillan と Aarseth によって実装された。[MA93a] もう一つは、実際にツリーから一旦問題の粒子を取り除き、もういちど付け加えるという方法である。著者はこの方法を実装したことがあって、たいして面倒ではないが現在のところ広く使われてはいない。

現在のところ、ツリーと独立時間刻みとの組み合わせはシミュレーションのカバーする時間範囲（系のダイナミカルな時間スケールで測った）が非常に短く、そのかわり扱う粒子数が多い宇宙論や銀河形成のような問題、あるいはあまり空間構造が発達しない惑星形成のシミュレーションのような場合に限って使われている。球状星団における自己相似的な密度上昇のような、系のわずかな部分だけが極端に短い時間スケールを持つという場合に、どのように並列化をすればよいかがまだほとんどわかっていないからである。

可能性としてはうまくいくかもしれない方法としては、 Springel らが導入した並列化の手法がある。[SaSDW00] 彼らの方法では、前に述べたように ORB で空間分割して粒子をば

らまくが、あるタイムステップで動かす粒子への力を計算するときに、その粒子があるプロセッサが他のプロセッサから必要になる情報をもらってきて重力を計算するのではなく、逆に動かす粒子の座標の方を他のプロセッサにおくって力を計算してもらい、力を送り返してもらって合計するという方法をとる。力の計算が分散しているぶんだけ、粒子があるプロセッサがその粒子への力の計算を全部やる方法に比べれば効率が上がる可能性がある。

とはいえ、もともと独立時間刻みを効率的に並列化するだけでわりあい面倒なので、ツリー法が付け加わるともっとうまくいかなくなるのはある意味やむをえないところがある。

結局のところ、並列化がうまく出来て効率をあげられるためには、もとの問題に十分な並列性がある必要がある。ところが、独立時間刻みにしてもツリー法にしても、単純なやりかたに比べて計算量を減らしてはいるがその分だけ並列性も減っているわけである。

もっとも、粒子数があまり大きくない問題に対してツリーと独立時間刻みを組み合わせる方法の研究があまり進んでいない理由はもうすこし違うところにあるかもしれない。一つは、現在扱える程度の粒子数 (数万以下) に対してはツリー法よりも前に述べたネイバースキームのほうが有利であり、またこちらのほうが並列化についても (おそらくは) 容易であることである。粒子数が数万程度の時、ネイバースキームでは計算精度をほとんど落さないで直接計算の10-20 倍程度加速できる。これに対してツリー法では計算量 1/10 に程度にするには精度をかなり落す必要がある (具体的には力の精度で 10^{-4} 程度)。それで得た答が正しいかどうかが良くわからないので、あまり使いたくないわけである。

もう一つは、専用計算機の存在である。これについては次章でもうすこし詳しく述べる。

というわけで、ここまでで重力多体系の数値計算を普通の汎用計算機でやるという話は一応おしまいである。

8 別のアプローチ — 計算機を作る

ここでは、大規模な計算を可能にするためのこれまで述べてきたアルゴリズムの改善とは相補的なアプローチ、つまりハードウェアの計算速度そのものを速くするという方向について簡単に述べたい。原稿依頼ではあんまりこの辺の話は、、、とか書いてあったような気もするが、多体シミュレーションが数値実験であるとすれば、ソフトウェアと同様にハードウェアも実験道具であり、ソフトは自分で作るのにハードは買ってくるというのもある意味奇妙な話ではないだろうか。

まあ、といっても、最近はソフト自体も大規模化し、個人、あるいは小さな研究グループの手に負えなくなってきている。例えば、FMM のような面倒なアルゴリズムを、分散メモリ型並列計算機のようなプログラミングの手間が非常に大きい計算機の上に実現し、しかもある程度まともな実行効率を得るのは容易なことではない。

そういうことを考えると、ソフトだけでも大変なのに、ハードまで自分で作って、さらにそのハード用のソフトウェアまで自分で作るなんていうのは全く論外である、、、と思われる方が多いのではないだろうか。

というわけで、以下必ずしもそうでもないのではないかという話をしたい。

8.1 基本的発想

多体問題シミュレーションにおいて、計算時間の大半というよりも 99% 以上を占めるのは、粒子間の相互作用の計算、つまり運動方程式の右辺の評価である。従って、計算速度をあげるためには、何らかの方法で右辺を速く評価できるようにするか、あるいはいろんな方法で右辺を評価する回数自体を減らすことになる。ここまででは、計算機は与えられているものとしてこれらをどうやって実現するかという話をしてきたわけである。

しかし、考えてみると粒子間の相互作用の計算というのは極めて単純な式の繰り返し計算である。高級言語のプログラムで書けば数行だし、アセンブラでも 1 ページくらいにしかならない。汎用計算機を使って多体シミュレーションをしているときは、ほとんどの時間汎用計算機をこの単純なプログラムを走らせるためだけに使っているわけである。

このような状況では、粒子間の相互作用の計算だけを高速にやる補助回路を計算機につけてやればシミュレーション全体を高速化できる。これは、例えばゲーム用のコンピュータなどで、画像処理等のためにいくつかの専用回路をつけてそういった処理は CPU を使わないで済ませるというのと考え方としては同じである。

しかし、多体問題用の専用回路には、実は例えば画像処理用回路に比べて原理的に有利な点がある。画像処理の場合、3 次元の特殊な処理でなければ処理のための計算量はデータ量に比例する。従って、処理速度をあげるためには、データ転送速度も同時にあげる必要がある。このために、CPU やメインメモリと画像処理 / 表示部分とを非常に高速なインターフェースでつなぐ必要が出てくる。高速にしないといけないということは、現在の計算機の構成では CPU の「近く」に持ってくる必要があるということであり、そのためには計算機の構成自体がそういうものを考慮して作られている必要がある。例えば AGP バスがその例ということができる。

これに対し、粒子間の相互作用計算では、単純な計算法では計算量が粒子数 N の 2 乗に比例する。つまり、データ量の 2 乗に比例して計算量が増えることになる。もちろん、ツリー法などではそううまくはいかないが、それでもデータ量に対して計算量が非常に多いということには変わりがない。このために、それほど高速ではない汎用の I/O バスを通して接続してもまあまあの性能がでる。例えば、最近の大抵の計算機で採用されている PCI バスを使った接続でもまあまあの性能を出せる。

そういうわけで、概念的には、粒子系シミュレーション用の専用計算機は図 13 のような構成をとることになる。

専用回路の側は相互作用を計算するだけである。我々はこちらの部分に GRAPE (GRAvity PipE) という名前をつけている。とりあえず、もっとも単純な直接計算で時間刻み一定の場合を考えると、専用計算機としては例えば系の全粒子を受けとって自分のメモリにしまい、それから 2 重ループで全粒子への他の全粒子からの力を計算する。最後に求まった力を送り返す。

とはいえ、これまで長々と話をしてきたように、直接計算で時間刻み一定しか出来ないのではいかに専用回路が高速であったところで出来ることは制限される。例えば、現在の汎用並列計算機で最高速のものは 10Tflops 程度の性能を持つが、これでも粒子数が 10^7 になれば 1 ステップ 5 分、10^8 なら 10 時間となってすでに現実的とはいい難い。このために、

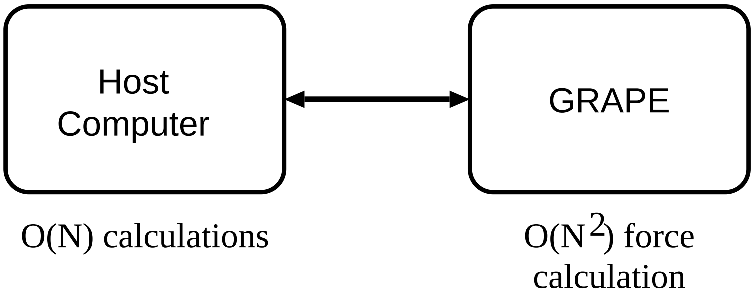

図 13: 粒子系シミュレーション用の専用計算機の基本構造

専用計算機といっても実際に意味がある計算をするためにはこれまでに述べてきたような数値計算法、特に独立時間刻みやツリー法との組み合わせが重要になる。これらについては次節以降で扱うとして、ここではとりあえず専用回路にすることでどの程度高速になりえるかということを考える。

さて、専用回路にすることで高速化出来るのはそもそもなぜであろうか ?そんなのは当たり前であるとなんとなく思っているだけでは、実際にどういうふうに回路を作ればどれだけ高速化出来るか、またそもそも高速化の理論的な限界はどこかといった問題に答えることが出来ない。それでは、何か作ったとしてもそれに意味があるものかどうか評価出来ないのである。

というわけでまた余談になるが、並列計算の場合でもそうだが、多くの研究というか並列プログラムの開発で、そこで実装することにしたアルゴリズムがネットワークの速度やプロセッサ台数が与えられた時に、もっともうまくいってどの程度の性能がでるかということは普通は理論的に評価できる。理論的にどう振舞うのかわからないほど複雑なアルゴリズムではうまくいく気遣いはないからである。しかしながら、著者の知る限りではそのような分析をやってから仕事を始めた、あるいは途中でもちゃんとそういう検討をして方針を考えたように見えるプロジェクトはあまり多くないというかほとんどない。このような状況ではアルゴリズムの進歩はいいところメトロポリス・モンテカルロ法的なものでしかないわけである。

もちろん、計算機科学の専門家が書いた論文の場合には、少なくとも実装の性能についての評価はそれなりにちゃんとしている。しかし、その場合でもアルゴリズムの性能の理論的な評価はほとんどされていないことが多い。なぜ並列計算法の研究が、そのような理論不在の状況にあるのかということは著者には大きな謎である。

と、話を戻す。高度なアルゴリズムと組み合わせた時の話は後でするので、ここではまずは単純な直接計算で時間刻み一定の場合を考える。

普通の計算機、具体的には普通の PC とかそのクラスタの場合には、重力計算は汎用 CPU で実行される。この時に実現可能な速度は、どんなに良くても CPU の理論ピーク性能を超えることはあり得ない。例えば、 750 MHz の EV6 Alpha CPU では、あらゆることがうまくいったとしてクロックサイクル毎に実行できるのは加算と乗算が一つづつである。最近の x86 系の CPU である AMD Athlon や Intel P4 でも事情は同じようなものである。

表 1: 浮動小数点演算器に必要なトランジスタ数

演算器の種類	トランジスタ数
倍精度乗算器	1×10^5
単精度乗算器	2×10^4
倍精度加算器	3×10^4
単精度加算器	1×10^4

さて、普通に2粒子間の相互作用を計算するのに必要な演算の数を数えると、乗算と加算が合わせて19(だったかな)、平方根が1、割算が1になる。平方根や割算にどれくらいの時間がかかるかはCPUによるしまた演算ライブラリの出来によっても変わる (細かいことをいい出せば、要求する精度にもよる。例えばニュートン法反復で平方根や割算を実現する場合には単精度なら倍精度に比べて一回反復数がすくなくてもいいはずである。) が、まあここではとりあえずそれぞれ10演算程度としておこう。すると、あらゆることがうまくいったとして1ペアの重力を計算するのにはこれらのCPUでは30クロックサイクル程度かかることになる。

まあ、実際にはいろいろ無駄があるのでこの2-3倍はかかるが、とりあえずこれよりも速くはなり得ないという上限はこのあたりになる。1 GHzのCPUを使った場合で1秒に3千万ペアを評価できることになる。

さて、ここで、「上限」として、「CPUを作るのに使っているすべてのトランジスタを演算器に使うことができ、それらがすべて並列に動作したとしたらどれくらいのことが出来るか」というのを考えてみる。現在のデジタル回路を使った計算機である限り、これ以上には速くできないのは明らかであろう。

現在の典型的なCPUで使われているトランジスタ数は1-2$\times 10^7$である。これに対し、浮動小数点演算器を構成するのに必要なトランジスタの数は、パイプライン段数とか細かいことで多少は変わるが、大雑把にいって表1の程度である。

すべての演算を倍精度で行なう意味はほとんどない。というのは、すでに見てきたように軌道の指数的発散の問題があるし、これに影響するのは近傍の粒子との相互作用、すなわち桁落ちがもっとも大きく効くところである。従って、最初の座標の引き算と、最後の力の累算は倍精度で行なうとしても、それ以外は単精度で行なうというのがまあ妥当な選択であろう。これよりも精度を落してもほとんどの問題であまり問題はないが、しかしそれほど劇的には演算器が小さくならないからである。なお、理論的には演算器の大きさは乗算器の場合仮数のビット数の2乗に比例するが、仮数が小さくなると指数のところの回路の大きさが無視出来なくなる。加算器の場合は、高速な加算器やシフタを使った時に大きさがビット数pの$p \log p$に比例する。

まあ、ここでは、倍精度乗算器を使わないとすれば典型的な演算器の大きさは2×10^4トランジスタと思っておいていいであろう。ということは、今の典型的なCPU程度のLSIには最大で500もの演算器が入るということである。これが1 GHzのクロックで動くとすれば、チップ単体での性能はピークで比べて汎用のCPUの250倍ということになる。こ

の 250 倍というところに、専用回路の潜在的なアドバンテージがあるということになる。

さて、専用回路 (LSI) を作った時の上限の性能がこれくらいであったとして、このアプローチが現実的なものであると主張するためには以下のような疑問に答える必要があろう。

- 本当に 500 個も演算器があるチップを作れるのか ?
- チップを作るのにはどれくらいお金が掛かるのか ?特に初期費用と量産コストのうちわけはどんな感じか ?
- お金以前に、誰がそんなチップを設計してくれるのか ?
- チップでは 250 倍になったとしても、それだけなら PC クラスタとかで沢山 CPU を持ってくればそっちのほうが速いのではないか ?

以下、順に見ていくことにする。

まず、実際に 500 も演算器があるチップを作れるのかということだが、これは以下の 3 つの手法の組み合わせによって可能になる。第一は、相互作用計算自体をパイプライン化したハードウェアを作ることである。つまり、相互作用計算の演算の手順通りに演算器を並べてしまうのである。概念を図 14 に示す。このパイプラインは基本的にクロックサイクルごとに新しい粒子データを一つ受けとり、パイプラインをずーっとデータが流れていって求まった相互作用を最終段のアキュムレータで集計していく。各演算器が何をするかは始めから決まっているので、このパイプラインのなかで制御するべきものは最終段のアキュムレータだけである。これは、計算を始める時にデータをクリアするのと、それから計算が終ったところで結果を固定するというだけで、電子回路としては標準的なロードとクリア機能がついたフリップフロップでアキュムレータのレジスタを構成して、それを外からの制御信号で制御するというだけである。このパイプライン化によって、30 個程度の演算器を集積することができる。エルミート公式を使うために時間導関数も計算することにすれば、50 個程度の演算器が集積可能である。このパイプラインの基本的な使い方としては、外づけのメモリとメモリアドレス生成回路 (といっても、実質は単なるカウンタ) と組み合わせて全体回路を構成する。計算の手順としては、最初にメモリに力を及ぼすほうの粒子を書き込み、それからチップ内の力を受ける方の粒子座標レジスタに値を書き込み、カウンタを回して計算させ、終ったところでアキュムレータから値を読み出す。これを、力を受ける粒子の数だけ繰り返す。

第二は、複数パイプラインを同一チップに集積することである。複数のパイプラインは、それぞれ別の粒子への力を計算することにしておく。こうすれば、計算中にチップに供給する必要があるデータ量はパイプラインの本数に無関係になるので、沢山パイプラインを入れても効率が下がることはない。これは通常の計算機では並列化が一般には難しく、そのために半導体技術が進んで大量のトランジスタが集積可能になっても、チップに集積された演算器の数がほとんど増えていないのとは大きな違いである。もちろん、これは使い道を限って並列性が高いものにしているからである。

第三は、我々が仮想マルチパイプラインと呼んでいる方法である。この方法では、一つのパイプラインを時分割で使い、複数の粒子への力を計算する。なぜこんなややこしいことをするかというと、メモリからのデータ転送の必要バンド幅を減らすためである。つまり、一つのパイプラインが複数の粒子への力を計算するということは、メモリからの粒子デー

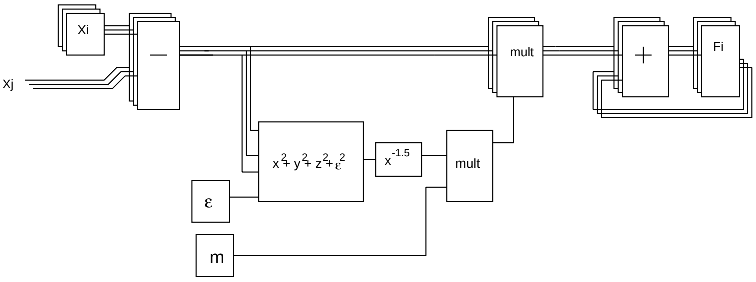

図 14: 重力計算用パイプライン

タはその分だけ同じものを使い回せるということになる。バンド幅がどの程度になるかをここで紹介しておくと、位置を 64 ビット、質量を 32 ビットで表現するとして、パイプラインのクロック速度が 100 MHz とすると、単純な設計では転送速度が 2.8 GBytes/s となる。これは最新のマイクロプロセッサでのプロセッサとレベル 2 キャッシュのデータ転送速度に相当し、実現するのは不可能ではないがかなり特殊な技術を使った実装が必要になる。マイクロプロセッサのように大量に製造する場合には量産効果でコストを下げられるが、数が少ないという前提の下ではあまり特殊な技術を使うのは現実的ではない。ここで、仮想マルチパイプラインの多重度を例えば 8 とすれば、データ転送速度は 350MB/s 程度となってそれほど難しい話ではなくなる。パイプラインクロックが 100 MHz というのは最近のマイクロプロセッサの 1.5 GHz とかいうのに比べてずいぶん低いではないかと思うかもしれない。これは純粋にチップ設計にどれだけの時間、人とお金を掛けられるかによっている。

次に、お金の話である。チップを作るのにはかなり莫大なお金が掛かるのは確かである。が、上でもちょっと出たがあまり極限的な設計をしなければマイクロプロセッサで掛かるような 100 億円単位というわけではない。我々が作ったものの例をあげると、GRAPE-4 の場合で試作コスト (サンプルチップまで) が 2500 万円、GRAPE-6 ではだいぶ高くなって 1 億ちょっとになっている。これは半導体技術が GRAP-4 の 1μm から 0.25μm と進歩した分初期費用が上がったためと、LSI の詳細設計を基本的には外注にしたためである。量産時のチップ単価は GRAPE-6 で 3 万円程度で、まあまあ我慢出来る範囲であろう。

次に、誰がそんなチップを設計してくれるのか ?ということだが、これは、結局、予算さえあればどこでもやってくれる。仕様書レベルのものから論理設計を全部やってもらうとしても、論理設計と検証は優秀なエンジニアが一人でやったとして一年程度で出来る。優秀なエンジニア一人、1 年間のコストというのがどれくらいかというのは評価が難しいところだが、大企業に頼むとまあ 1 億かもうちょっと高いあたりになろう。これはもちろんそのエンジニアがそんな給料をもらっているというわけでは (日本の企業では) ない。が、まあ、優秀なエンジニアあたりの企業のトータルコストというのは、良くてこれくらいであろう。悪いともっとかかる。

とはいえ、これは、仕様書と検証用テストデータおよび出力はこちらで準備するとしての話である。つまり、論理設計そのものは自分ではやらないとしても、パイプラインやその

まわりの制御回路の動作仕様はこちらで決める必要がある。とはいえ、高エネルギー実験や天文観測の研究グループが日常的にやっているレベルの回路設計・開発に比べればたいして難しいものではないので、そのあたりの知り合いとかがいれば教わりながら、、、というので出来る程度の話ではある。

もちろん、これはある程度というよりは巨額な予算をとってきた場合の話である。読者のなかでこの記事を読んでそういうことをやってみるのも悪くはないと思う人がひとりくらいはいたとして、予算が1億円からないと始まらないというのではもちろん話にならない。

我々が専用計算機を作ろうというのを始めたのは1988年で既に13年も前のことになる(なんか嫌な感じ)。当時は、半導体技術、基本的には集積度のレベルが今とは2桁以上違っており、例えば完全パイプライン動作の64ビットの浮動小数点演算器を1チップで実現したLSIは存在しなかった。この時点では、市販の浮動小数点演算LSI、あるいは固定小数点のALUやROMをつかったテーブルルックアップを組み合わせて演算パイプラインを組み上げて、ラッピングボード1枚で当時の汎用マイクロプロセッサの100倍以上の性能を実現することができた。当時の汎用マイクロプロセッサの100倍以上の性能というのは、いうまでもなく現在のマイクロプロセッサ程度の性能ということであり、このような市販の演算用LSIを組み合わせる方法ではそれ以上の速度を実現するのは難しい。これは単にボード上にLSIを並べた時にその間のデータ転送を速くするのは大変だからである。また、それ以前に汎用の演算LSIというもの自体が軍用の特殊なものなどを除いてほとんど存在しなくなっている。もともとは演算LSIを使って作られていた回路のほとんどが、高速を必要とする場合には専用のカスタムLSI、半導体技術の進歩によってそれほどの速度を必要としなくなった場合には汎用のCPUやDSPによって置き換えられたからである。もともと、このような演算専用LSIのかなり大きなマーケットはマイクロプロセッサに付加する浮動小数点演算ユニットであった(WTL 1264など)。マイクロプロセッサが浮動小数点演算回路も集積した時点で、マーケットのほとんどが失われたのである。

従って、現在において小規模の専用計算回路を設計・開発し、しかも意味がある程度の性能を実現するのは10年前に比べてある意味でははるかに困難になっている。10年前は、一つのWindow of opportunityであったことは間違いない。というのは、さらにその10年まえ、つまり20年前を考えると、演算器を並べて専用演算回路を作る以前に、4ビットとか8ビットの乗算器LSIを並べて乗算器を作る必要があったわけであり、ある程度複雑な専用演算回路を作り上げることは不可能ではないもののかなり大がかりで費用も人手もかかるプロジェクトとなったからである。

さて、では現在ではもうどうしようもないのかというと、これは実は必ずしもそうではない。FPGA(Field programmable gate array)というものが半導体技術の進歩につれて順調にその規模、速度を拡大してきているからである。FPGAというのはその名前の通り、基本的には後から論理回路を書き込む(プログラムする)ことのできる汎用のLSIといってよい。どうやってプログラム可能にしているかというと、基本回路には小さなSRAMを使い、テーブル参照によって任意の論理式を実現する。さらにその基本回路間をまたプログラム可能なスイッチと配線マトリックスでつなぐことで、小さな基本回路では実現不可能な大規模・複雑な論理を実現するわけである。

プログラムするべきハードウェア回路は、通常のカスタムLSIと同じようにVHDLなどの

ハードウェア記述言語によって設計できる。多数ビットの加算器や乗算器は、多くの場合にFPGAメーカーが提供するVHDLコンパイラによって自動的に最適な回路構造に変換される。(速度優先か回路サイズ最小化優先かをある程度の範囲で指定できるものが多い)

回路規模でいうと、現在はメーカー公称で100万ゲートを超えるものが入手可能(高価ではあるが)になっている。ここでゲートはカスタムLSIやFPGAの大きさを表示する時の伝統的な単位で、NAND回路一つ分を1ゲートと数える。トランジスタ数としてはCMOSの場合4トランジスタにおおむね相当する。動作速度も、設計に注意を払えば100 MHz近くまで到達できなくもない。

まあ、100万ゲートというのはメーカーさんの主張で、実際にはその半分程度の規模の回路が入れば万歳だが、それでも200万トランジスタ、したがって単精度なら浮動小数点演算器が100個程度入ることになる。これは上に書いたカスタムLSIの数字に比べるとかなり小さいが、これはもちろんプログラム可能性のために無駄が発生するからである。逆に、その差が5倍程度なのは、一つにはカスタムLSIの数字が少し古いからではある。この原稿を書いているのは2001年2月である。FPGAの数字は現時点のものだが、カスタムLSIの数字は我々が実際にLSIを作った1998年時点のものになっている。そのためにFPGAのほうが技術進歩の分の数倍の得になっている。もう一つは、カスタムLSIのほうは技術的な極限まで大きいものを考えてはいないからである。

100 MHzで演算器が100個だと10 Gflopsにしかならないわけで、現在の最高速のマイクロプロセッサに比べて10倍しか速くないということにはなる。しかし、それでも以下に述べる理由からマイクロプロセッサを10個並べるよりは有利になる可能性が高い。一つは、カスタムLSIの場合と同様に、100個の演算器を使ってパイプラインを構成し、繰り返し演算の間はそのすべてを有効に使うというのが比較的容易に出来るからである。さらに、チップ1-2個のシステムで実証が出来れば数百万円の予算で100チップ程のシステムが構成出来るであろう。これはマイクロプロセッサ1000台に相当し、例えば通常の大学の研究室レベルでは購入予算もそうだがスペース・消費電力からも実現困難なシステムになる。

さて、もちろん、FPGAでは試作コストは掛からないといっても、中身を設計しないといけないのはカスタムLSIと同じである。固定小数点の演算器程度ならばコンパイラが作ってくれるし、ソフトウェアを買ってくれば浮動小数点演算器もコンパイラが作ってくれるようになる。従って、どういう計算をさせたいのかこちらにわかっていれば、中身の設計は(面倒ではあるが)それほど難しいという話ではない。

これはもちろんカスタムLSIの場合でも全く同じことであるが、むしろ面倒なのは次の2つ、つまり一つは動作検証、もう一つは周辺回路の設計である。

動作検証は、要するに「この回路でいい」かどうかを確認するわけである。例えば、データ形式を普通の64ビット浮動小数点と決めてしまえば、回路規模は大きくなるが検証は簡単になる。普通にPCとかで計算した答と合わせるだけだからである。しかし、これは多くの場合に回路が大きくなりすぎるので、やはり普通とは違うデータ形式を使いたくなる。例えば32ビット浮動小数点演算を使うなら、64ビット型との変換をちゃんとやればやはり普通にPCとかで計算した答と合わせるだけで済み、検証は比較的簡単である。

とはいえ、実際にはいろいろ悩ましいことはある。大抵の場合に全演算に32ビット精度が

いるわけではないし、またそもそも浮動小数点がいるところは少ない。こういったところを、必要に応じて最小限のデータ長や形式を選んで実現していくことによって、全体を 64 ビットでやったりあるいは 32 ビットしか使わなかったりする場合に比べてかなり回路規模を小さく出来る。しかし、そのためにはそのような演算器を作らないといけないし、もっと大変なことにそのような演算器を使って作ったパイプラインからでてくるべき答を知らないといけない。さらに、その精度が必要を満たしているかどうか、つまり、そのパイプラインでシミュレーションしてまともな答が出るかどうかもチェックする必要がある。

とはいえ、FPGA の場合は、これらすべてを実際に回路を作ってからやるというのも可能ではある。カスタム LSI の場合には作り直しには莫大なお金と時間が掛かるので、十分に時間をかけ、まず汎用計算機上でパイプラインのシミュレータをつくってそれを単体テストし、さらに実際の N 体シミュレーションプログラムに組み込んで評価する。その評価が終った後で、実際のハードウェアを設計しその動作がシミュレータと同じかどうかをテストデータで検証するというかなり回りくどい手順が必要になる。この検証に、設計よりもはるかに長い時間がかかるといっても過言ではない。FPGA ではこのへんはある程度いい加減にやれる。

周辺回路のほうは、設計自体が面倒ではある。例えば、FPGA の石を買ってきたとして、それに少なくともメモリをつけないといけないしホスト計算機に組み込むためのインターフェースもいる。これらをまとめてプリント基板を設計し、どこか業者に製造を依頼しないといけない。

とはいえ、例えば標準的な PCI インターフェースと外づけメモリに大規模 FPGA を組み合わせたようなものならば、国内でもいくつかの企業が商品化しているものがある。そのようなものを使ってみるのがもっともハードルが低い方法であろう。

最後に、並列化について簡単にまとめる。LSI 一つでマイクロプロセッサの 100 倍以上の性能が実現できるとしても、一つしか使えないならマイクロプロセッサ 100 個、というよりは PC を 100 台並べたクラスターの方が速いことになる。もちろん、既に見たように重力多体問題に対して 100 台を超える規模の PC クラスタで十分な性能を出した例というのはあまりないが、まあ、原理的にはそういうことではある。

重力多体問題専用計算機、というより相互作用計算専用回路の利点は、かなり大規模な並列性が比較的容易に実現出来るということである。もちろん、これはやりたい計算自体に高い並列性があるからである。アルゴリズムによっては次節以降に述べるようにいろいろややこしい話がでてくるが、単純な直接計算で一定時間刻みの場合には、数万から 100 万以上に及ぶ粒子から、またそれだけの数の粒子への力を計算するわけで、力を受けるほうはすべて独立、つまり並列に計算出来るしまた力を及ぼすほうも、最終的に合計するところの計算時間が問題にならない範囲では並列に計算できる。

この並列性は既に単一のチップの話をする時に利用しているわけで、パイプライン化できるのは力を及ぼす方がある程度多いからであり、また複数パイプラインを並列動作させられるのは力を受ける側が複数あるからである。パイプライン段数はせいぜい 100 段程度なので、粒子数が 100 万とかいっているときにはほとんど無視できる。従って、一つの粒子への力をさらに 100 とか 1000 チップにわけで、並列に計算してもパイプライン遅延は問題にならない。もっとも、1000 もパイプラインがあるとそれを順番に合計していったので

は時間が掛かり過ぎるが、これはトーナメント方式で合計すれば2入力の加算器を使ったとして10段、4入力のものを特別に設計すれば5段のツリー構造ネットワークで済む。さらにこの全体がパイプライン化できるので、段数が多くても性能にはそれほど影響しない。こういった回路は、規模的には現在のFPGAに容易に収まる。

複数粒子への力を並列に計算するほうはもっと簡単で、単にメモリとアドレス生成ユニットの回路に複数のチップをぶら下げるだけである。

適当な大きさのプリント基板を作れば、チップ数十個くらいは載せられる。これで、普通のPCクラスタとしては1000台程度の規模(大規模な計算機センターにある程度)とほぼ同等の性能が実現できる。それ以上の数を並べたければ、プリント基板を複数並べればいい。

並べたものをどうやって使うかというような話になると、段々問題というよりもアルゴリズム固有の側面が大きくなる。これらは以下の節でもうちょっと議論する。

8.2 独立時間刻みと専用計算機

すでに繰り返し述べてきたように、専用計算機がいかに高速であっても、ある程度賢いアルゴリズムが使えなければ実用的な意味はほとんどない。まず、ここでは、独立時間刻みについて考える。

独立時間刻みのために必要な機能を専用計算機に作り込むのは実はそれほど大したことではない。普通の場合となにが違うかというと、単に重力を計算する時に位置がそのまま使えるわけではなくて、予測子を評価してから重力を計算するというだけである。予測子は単なる時間の多項式なので、汎用計算機で計算するのと全く同様にホーナーの公式を使って高い次数から順に計算していけば良い。

とはいえ、一つ問題なのは同時に積分できる粒子の数である。もともとの独立時間刻みでは一度に一つの粒子しか積分しない。これでは、多数パイプラインがあってもそれらを一つの粒子への重力を計算するのにしか使えないことになる。但し、すでに並列化のところで述べたような、ブロックステップ法を使えばそこそこの数の粒子を並列に積分出来るので、これはあまり大きな問題ではない。[Mak91a]

我々が開発したシステムの中では、GRAPE-4[MTES97]およびGRAPE-6が独立時間刻み用(といっても、もちろんそうでない計算コードでも使えるが)に設計されている。GRAPE-4は1992年に開発を始めて1995年に完成したシステムであり、カスタムLSI一つ(HARP chip)に重力とその時間導関数を計算するパイプラインを入れ、もう一つのカスタムLSI(PROMETHEUS chip)に予測子計算を入れた。HARPチップは2粒子への力を仮想マルチパイプラインにより並列に計算する。

GRAPE-4では、プリント基板にはPROMETHEUSチップ1つとHARPチップ48個を載せた。計算中はHARPチップはPROMETHEUSチップからすべて同じデータを受けとるので、$48 \times 2 = 96$個の粒子への力が並列に計算されることになる。GRAPE-4全体はこのボード(プロセッサボード)36枚からなる。これを9枚ずつラックに入れ、それらをコントロールボードと呼ばれる別の基板から制御する。プロセッサボードで求まった

力はコントロールボード上でハードウェアロジックで加算され、4 枚のコントロールボードで求まった力は最終的にホスト計算機上のソフトウェアで合計する。GRAPE-4 のピーク速度はおよそ 1.1 Tflops に相当する。なお、動作クロックは 32 MHz である。

GRAPE-6 は 1997 年に開発を始め、現時点で小規模なシステム (4 Tflops 相当) が動作しており、大規模システムの組み立て中である。GRAPE-6 では、GRAPE-4 に比べてはるかに大きな LSI が利用できるようになったため、1 チップに重力計算パイプライン 6 本と予測子パイプラインを組み込んだ。一本の重力計算パイプラインは 8 本の仮想パイプラインとして動く。これにより必要なメモリバンド幅を減らしている。その代わりに、1 チップですでに 48 粒子への力が同時に計算されてしまう。複数チップでこれがさらに増えるのは嫌らしいので、チップごとにメモリをつけて、ボード上でも複数ボード間でも違うチップからの力は合計する。但し、現在のところ普通に合計することを考えているのは最大 512 チップまでで、それ以上はチップ数に比例して並列に計算される粒子数が増えるような使い方をする予定である。

なお、予定としては 4096 チップを並列動作させ、130 Tflops を実現したいなあということになっているが、予算的にちょっと苦しい。

さて、独立時間刻みができれば次はネイバースキームも、、、というのが人情であるが、いまのところ良い方法は見つかっていない。問題なのは、ネイバーリストが粒子ごとに違うということである。このために、ネイバーからの力を並列計算するためには違う粒子への力を計算するパイプラインは、力を及ぼす粒子も違うということになる。このために、ネイバーからの力を高速に計算することが出来るハードウェアは、どのように設計しても本質的にメモリバンド幅が大きくなる。ところが、メモリバンド幅は製造の困難さに大きく影響するので、そういうものをつくるのは難しいわけである。

8.3 ツリー・FMM と専用計算機

ツリー法や FMM でも、計算量の多くを占めるのは粒子同士の相互作用の計算である。まあ、高次の多重極展開を使うと単純には粒子同士の相互作用ということにはならないが、既に述べた疑似粒子法を使えばツリー法の場合は結局計算するのは粒子からの重力になる。FMM の場合でも、もっとも計算量の多い多重極展開を局所展開に変換するところは、疑似粒子法とアンダーソン法を組み合わせることで粒子からの重力を計算するだけで済むようになる。以下、主にツリー法の場合の話をする。FMM は専用計算機がどうこうという以前にそもそも使われていないからである。

実際にはツリー法と専用計算機を組み合わせて高い性能を出すのにはもうちょっといろいろ工夫が必要である。ネイバースキームの場合と同様に、ツリー法では重力を受ける粒子ごとにすべて違った計算をする。しかも、ある意味ネイバースキームよりもさらに面倒なことに、どのような粒子 (節点) からの力を受けるかはあらかじめ決まっているわけではなくて、実行時にツリー構造をたどっていくことできまる。そういうものをそのままハードウェア化するのは、不可能ではないがやはりパイプライン 1 本で粒子一つしか計算出来ないとか、そういう問題が起きる。

実はベクトル化や並列化の場合でも、規模は小さいながら類似の問題が起きていて、その問題に対応するようなアルゴリズムも提案・実現されている。ここでの問題は、ベクトル化は出来るにしてもベクトル化されたループの中に条件分岐（ツリーを降りるか降りないか）があり、またメモリアクセスも連続ではない（ツリー自体はポインタを通してアクセスされる）ためにどうしてもハードウェアのピーク性能からみると低い性能になってしまうということである。典型的には、ベクトル計算機の場合理論ピーク性能の 10% 程度の性能になってしまう。

仮にある程度計算量が増えるにしても、力を計算するループのなかから条件分岐や間接アクセスを追放できれば理論ピーク性能の半分以上は出せるので、そういう方法はないかというのが問題であった。これに対する一応の回答は、「あらかじめある粒子に力を及ぼす粒子とセルのリストを作っておいて、それを連続な配列にしまっておく」というものである。こうすれば、少なくとも重力を計算している間はハードウェアのピークに近い性能がでる。

とはいえ、これは問題をすり替えただけなわけで、じゃあリストは誰がいつ作るんだというのがもちろん問題である。リストを各粒子用に作るわけなので、そのための計算量は重力計算自体の計算量とたいして変わらない。こちらが今度は時間が掛かってしまう。

ここで Barnes が考えたのは、リストのほうを複数の粒子で使い回そうということである。もちろんこのためにはリストが複数の粒子のどれにとっても良いものでないといけない。このためには、リストを作る時のツリーをたどる判定条件をすこし厳しくして、複数の粒子のどれか一つについてでも条件が満たされなければ下におりるということにすればよい。

実際には物理的に近くにある複数の粒子は、ツリー構造を使って適当な大きさのセルの中にある粒子ということにするのが簡単である。この時には、中の粒子を見なくてもセルの幾何学的な形状から判定条件を決めればいい。つまり、普通に相手との距離を計算する代わりにセル内の任意の点から相手への最短距離を求めればいい。

セルの中にある粒子数を大きくとると、ツリーをたどる回数はいくらでも減らすことができる。しかし、こんどはリストの長さのほうが伸びていくので、重力計算の計算量がだんだん増えてしまう。従って、どこかに計算時間が最小になるようなセルの大きさがあることになる。この最適点はもちろんツリーをたどる計算の速さと粒子間の重力を計算する速さの比によっていて、粒子間の重力計算が速いと粒子数を大きくしてツリーをたどる回数を減らすことができる。つまり、GRAPE のほうが速ければある程度まではホスト計算機側の仕事を減らせるということになる。

実際にどの程度の性能かというと、少し古いデータで恐縮だが GRAPE-5[KFMT00] 2 枚にホスト CPU として EV6 600 MHz の Alpha を組み合わせたもの [KFM99] で、Warren と Salmon による並列ツリーコードを 140 台の EV5 600 MHz Alpha 上で実行したもの [WGL$^+$98] の半分程度の性能が出ている。言い替えると、GRAPE-5 2 枚で Alpha 70 台分の性能ということになる。600 MHz EV5 Alpha というと若干旧式だが、名目ピーク性能では 1.2 Gflops で最近の Intel プロセッサとさして変わらない。さらに、Warren と Salmon のコードは非常に高速なものであり、他の同様なコードに比べて 2-3 倍は速く、理論ピークからそんなに遠いわけではない性能が出ている。

従来は、GRAPE では粒子からの力しか計算出来ないということで、モノポール（重心に

おいて、重力には符号がないので精度としてはダイポール）しか扱えず、高精度の計算では計算量が急速に増大するという問題があったわけだが、疑似粒子法によってその問題もほぼ解消した。

一層の高速化のためにはホスト計算機側を並列化する必要がでてくるが、これは原理的には既存の並列プログラムを手直しして GRAPE が使えるようにするという作業になる。これまで、そのような並列実行環境が実際にはなかったために現時点ではホストが並列で GRAPE を使うプログラムというものは存在しないが、この 1 月に国立天文台に 16 台の Alpha クラスタのそれぞれに GRAPE-5 をつける並列システムが導入されたことを契機に開発が進むことを期待している。これにより、従来は 1000 台規模の超並列システムでのみ可能であった 1 億粒子を超える大規模シミュレーションが、比較的小規模なクラスターで実行可能になる。

8.4 すべてを組み合わせる？

さて、専用計算機 GRAPE と独立時間刻みをうまく組み合わせることが出来ることはわかった。また、GRAPE とツリー法もまあまあうまく組み合わせてそれなりの性能を出すことができる。となると、このすべてを組み合わせることはできないかという希望が出てくることになる。ツリー法の項で述べたように、ツリー法と独立時間刻みを組み合わせるというのは一応出来ている話である。さらに GRAPE を使うのも、それぞれに対しては出来ている。ので、全部組み合わせるというのもなんとかなりそうだが、今のところうまくいっていない。まあ、単に誰もそんな方法のことを真面目に検討していないからというのが最大の理由なので、いろいろ考えればなんとかなるのではないかとは思う。まだ、ここで紹介出来るほどまとまってはいないが、基本的には実装できそうなアイディアはあるので 1-2 年のうちにはなんとか出来ると個人的には思っている。

9 まとめ

このノートでは、重力多体系の数値計算法を概観した。ここまで読んでいただけた方にはある程度伝わったかと思いたいが、数値計算法の開発・改良は多くの場合に重力多体系の進化を支配する物理過程についての理解とともに進んできている。計算機科学者・数値解析の専門家の寄与よりも、天文学の研究者の寄与のほうが圧倒的に大きいのはこのためである。

普通の考えでは「数値計算法」から外れるかもしれないが、著者が関わってきた専用計算機についてもその基本的な考えから独立時間刻みやツリー法などへの適用まですこし詳しく議論した。著者の信じるところでは、専用計算機の開発は数値計算法の開発と同様に、究極の目的である重力多体系の物理、あるいは天体現象の理解のための実験器具の開発である。その意味で、そのような実験器具の開発が可能である限りにおいて、開発を怠り出来合いの実験道具を買ってきてすましているのは研究者として「怠慢」であり、貴重な研究費をドブに捨てているようなものではないかという気もしなくはない。

と、ちょっと極端な意見かもしれないが、このノートはこれで終りなので勝手なことを書かせていただいた。また、機会があればどこかでお目に掛かりたい。

参考文献

[Aar63] Aarseth Sverre J. (1963) Dynamical evolution of clusters of galaxies, i. *Monthly Notices of Royal Astronomical Society* 126: 223–255.

[Aar85] Aarseth S. J. (1985) Direct methods for n-body simulations. In Blackbill J. U. and Cohen B. I. (eds) *Multiple Time Scales*, pages 377–418. Academic Press, New York.

[ADG+85] Applegate J. H., Douglas M. R., Gursel Y., Huner P., Seitz C. L., and Sussman G. J. (1985) A digital orrery. *IEEE Transactions on Computers* C-34: 822–831.

[And92] Anderson C. R. (1992) An implementation of the fast multipole method without multipoles. *SIAM Journal on Scientific and Statistical Computing* 13: 923–947.

[App85] Appel A. W. (January 1985) An efficient program for many-body simulation. *SIAM Journal on Scientific and Statistical Computing* 6: 85–103.

[AW86] Aguilar L. A. and White S. D. M. (1986) The density profiles of tidally stripped galaxies. *The Astrophysical Journal* 307: 97–109.

[BH86] Barnes J. and Hut P. (1986) A hiearchical o(nlogn) force calculation algorithm. *Nature* 324: 446–449.

[BHER95] Board J. A. J., Hakura Z. S., Elliott W. D., and Rankin W. T. (1995) Scalable variants of multipole-based algorithms for molecular dynamics applications. In *et al.* D. H. B. (ed) *Proceedings of the Seventh SIAM Conference on Parallel Processing for Scientific Computing*, pages 295–300. SIAM, Philadelphia.

[BS97] Blackston D. and Suel T. (1997) Highly portable and efficient implementations of parallel adaptive n-body methods. In *Proceedings of SC97*, pages (CD–ROM). ACM.

[Cha43] Chandrasekhar S. (1943) *Principles of Stellar Dynamics*. Dover, New York.

[CSS93] Calvo M. P. and Sanz-Serna J. M. (1993) The development of variable-step symplectic integrators, with application to the two-body problem. *SIAM J. Sci. Comput.* 14: 936–952.

[CSS97] Cano B. and Sanz-Serna J. M. (1997) Error growth in the numerical integration of periodic orbots, with application to hamiltonian and reversible systems. *SIAM J. Num. Anal.* 34.

[CTP95] Couchman H. M. P., Thomas P. A., and Pearce F. R. (October 1995) Hydra: an adaptive-mesh implementation of p 3m-sph. *The Astrophysical Journal* 452: 797+.

[EIN93] Emori H., Ida S., and Nakazawa K. (1993) New method for the numerical integration of an n-body system in an external potential. *Publications of the Astronomical Society of Japan* 45: 321–327.

[EJ96] Elliott W. D. and Jr. J. A. B. (March 1996) Fast fourier transform accelerated fast multipole algorithm. *SIAM Journal on Scientific Computing* 17: 398–415.

[ET99] Evans N. W. and Tremaine S. (October 1999) Linear multistep methods for integrating reversible differential equations. *The Astronomical Journal* 118: 1888–1899.

[Fuk98] Fukushima T. (1998) Symmetric multistep methods revisited. In *Proceedings of the 30th symposium on celestial mechanics*, pages 229+.

[Fuk99] Fukushima T. (1999) Symmetric multistep methods revisited: Ii. numerical experiments. In *Evolution and source regions of asteroids and comets : proceedings of the 173rd colloquium of the International Astronomical Union, held in Tatranska Lomnica, Slovak Republic, August 24-28, 1998 / edited by J. Svoren, E.M. Pittich, H. Rickman. Tatranska Lomnica, Slovak Republic : Astronomical Institute of the Slovak Academy of Sciences, 1999., p.309*, pages 309+.

[GR87] Greengard L. and Rokhlin V. (December 1987) A fast algorithm for particle simulations. *Journal of Computational Physics* 73: 325–348.

[GR88] Greengard L. and Rokhlin V. (1988) Rapid evaluation of potential fields in three dimensions. In Anderson C. and C.œ A. B. (eds) *Vortex Methods*, number 1360 in Lecture Notes in Mathematics,, pages 121–141. Springer-Verlag, Berlin.

[GR97] Greengard L. and Rokhlin V. (1997) A new version of the fast multipole method for the laplace equation in three dimensions. *Acta Numerica* 6: 229.

[GS86] Gurzadian V. G. and Savvidy G. K. (May 1986) Collective relaxation of stellar systems. *Astronomy and Astrophysics* 160: 203–210.

[Heg74] Heggie D. C. (1974) A global regularisation of the gravitational n-body problem. *Celestial Mechanics* 10: 217–241.

[Hen75] Henon M. (1975) Two recent developments concerning the monte carlo method. In *Dynamics of stellar systems*, pages 133–149. D. Reidel Publishing Co., Dordrecht.

[HHM93] Hernquist L., Hut P., and Makino J. (1993) Discreteness noise versus force errors in n-body simulations. *The Astrophysical Journal Letters* 402: L85–+.

[HK89] Hernquist L. and Katz N. (1989) Treesph – a unification of sph with the hierarchical tree method. *The Astrophysical Journal Supplement Series* 70: 419–446.

[HMM95] Hut P., Makino J., and McMillan S. (1995) Building a better leapfrog. *The Astrophysical Journal Letters* 443: L93–L96.

[HNNS78] Hachisu I., Nakada Y., Nomoto K., and Sugimoto D. (1978) Gravothermal catastrophe of finite amplitude. *Progess of Theoretical Physics* 60: 393–402.

[HO92] Hernquist L. and Ostriker J. P. (1992) A self-consistent field method for galactic dynamics. *The Astrophysical Journal* 386: 375–397.

[HS78] Hachisu I. and Sugimoto D. (1978) Gravothermal catastrophe and negative specific head of self-gravitating systems. *Progess of Theoretical Physics* 60: 123–135.

[HS96] Hardin R. H. and Sloane N. J. A. (1996) Mclaren's improved snub cube and other new spherical designs in three dimensions,. *Discrete and Computational Geometry* 15: 429–441.

[KFM99] Kawai A., Fukushige T., and Makino J. (1999) $9.9/mflops astrophysical n-body simulation with treecode on grape-5. submitted to SC99.

[KFMT00] Kawai A., Fukushige T., Makino J., and Taiji M. (August 2000) Grape-5: A special-purpose computer for n-body simulations. *Publications of the Astronomical Society of Japan* 52: 659–676.

[KM01] Kawai A. and Makino J. (2001) Pseudoparticle multipole method: A simple method to implement high-accuracy treecode. *ASTRO-PH* 0012041: –.

[KN96] Kinoshita H. and Nakai H. (1996) Long-term behavior of the motion of pluto over 5.5 billion years. *Earth Moon and Planets* 72: 165–173.

[KS65] Kustaanheimo P. and Stiefel E. (1965) Perturbation theory of kepler motion based on spinor regularization. *J. Reine. Angew. Math.* 218: 204–219.

[KYM98] Kokubo E., Yoshinaga K., and Makino J. (July 1998) On a time-symmetric hermite integrator for planetary n-body simulation. *Monthly Notices of Royal Astronomical Society* 297: 1067–1072.

[LE80] Lynden-Bell D. and Eggleton P. P. (1980) On the consequences of the gravothermal catastrophe. *Monthly Notices of Royal Astronomical Society* 191: 483–498.

[Lev06] Levi Civita T. (1906) Sur la r'esolution qualitative probl'eme des trois corps. *Acta Math.* 30: 305.

[LW68] Lynden-Bell D. and Wood R. (1968) The gravo-thermal catastrophe in isothermal spheres and the onset of red-giant structure for stellar systems. *Monthly Notices of Royal Astronomical Society* 138: 495–525.

[MA92] Makino J. and Aarseth S. J. (1992) On a hermite integrator with ahmad-cohen scheme for gravitational many-body problems. *Publications of the Astronomical Society of Japan* 44: 141–151.

[MA93a] McMillan S. L. W. and Aarseth S. J. (1993) An o(n log n) integration scheme for collisional stellar systems. *The Astrophysical Journal* 414: 200–212.

[MA93b] Mikkola S. and Aarseth S. J. (1993) An implementation of n-body chain regularization. *Celestial Mechanics and Dynamical Astronomy* 57: 439–459.

[MA96] Mikkola S. and Aarseth S. J. (1996) A slow-down treatment for close binaries. *Celestial Mechanics and Dynamical Astronomy, v. 64, Issue 3, p. 197-208.* 64: 197–208.

[Mak90] Makino J. (1990) Vectorization of a treecode. *Journal of Computational Physics* 87: 148–160.

[Mak91a] Makino J. (1991) A modified aarseth code for grape and vector processors. *Publications of the Astronomical Society of Japan* 43: 859–876.

[Mak91b] Makino J. (1991) Optimal order and time-step criterion for aarseth-type n-body integrators. *The Astrophysical Journal* 369: 200–212.

[Mak97] Makino J. (1997) Merging of galaxies with central black holes. ii. evolution of the black hole binary and the structure of the core. *The Astrophysical Journal* 478: 58+.

[Mak99] Makino J. (1999) Yet another fast multipole method without multipoles — pseudo-particle multipole method. *Journal of Computational Physics* 151: 910–920.

[McL63] McLaren A. D. (1963) Optimal numerical integration on a sphere. *Math. Comput.* 17: 361–383.

[McM86] McMillan S. L. W. (1986) The vectorization of small-n integrators. In Hut P. and McMillan S. (eds) *The Use of Supercomputers in Stellar Dynamics*, pages 156–161. Springer, New York.

[MH88] Makino J. and Hut P. (1988) Performance analysis of direct n-body calculations. *The Astrophysical Journal Supplement Series* 68: 833–856.

[MH91] Makino J. and Hut P. (1991) On core collapse. *The Astrophysical Journal* 383: 181–191.

[MIE90] Makino J., Ito T., and Ebisuzaki T. (1990) Error analysis of the grape-1 special-purpose n-body machine. *Publications of the Astronomical Society of Japan* 42: 717–736.

[MT98] Makino J. and Taiji M. (1998) *Scientific Simulations with Special-Purpose Computers — The GRAPE Systems.* John Wiley and Sons, Chichester.

[MTES97] Makino J., Taiji M., Ebisuzaki T., and Sugimoto D. (1997) Grape-4: A massively parallel special-purpose computer for collisional n-body simulations. *The Astrophysical Journal* 480: 432–446.

[PMHM01] Portegies Zwart S. F., McMillan S. L. W., Hut P., and Makino J. (February 2001) Star cluster ecology - iv. dissection of an open star cluster: photometry. *Monthly Notices of Royal Astronomical Society* 321: 199+.

[PMMH99] Portegies Zwart S. F., Makino J., McMillan S. L. W., and Hut P. (August 1999) Star cluster ecology. iii. runaway collisions in young compact star clusters. *Astronomy and Astrophysics* 348: 117–126.

[QKSL97] Quinn T., Katz N., Stadel J., and Lake G. (1997) Time stepping n-body simulations. *ASTRO-PH* 9710043: –.

[QT90] Quinlan G. D. and Tremaine S. (1990) Symmetric multistep methods for the numerical integration of planetary orbits. *The Astronomical Journal* 100: 1694–1700.

[SaSDW00] Springel V. and ane Simon D.M. White N. Y. (2000) Gadget: A code for collisionless and gasdynamical cosmological simulations. *ASTRO-PH* 0003162: –.

[SB83] Sugimoto D. and Bettwieser E. (1983) Post-collapse evolution of globular clusters. *Monthly Notices of Royal Astronomical Society* 204: 19P–22P.

[SB94] Skeel R. D. and Biesiadecki J. J. (1994) Symplectic integration with variable stepsize. *Annals of Numer. Math.* 1: 191–198.

[SB99] Spurzem R. and Baumgardt H. (1999) A parallel implementation of an aarseth n-body integrator on general and special purpose supercomputers. submitted to Monthly Notices of Royal Astronomical Society .

[SG92] Skeel R. D. and Gear C. W. (1992) Does variable step size ruin a symplectic integrator? *Physica D* 60: 311–313.

[Spi56] Spitzer Lyman J. (1956) *Physics of Fully Ionized Gases.* John Wiley and Sons, Chichester.

[Spi87] Spitzer Lyman J. (1987) *Dynamical Evolution of Globular Clusters.* Princeton University Press, Princeton, New Jersey.

[SSC94] Sanz-Serna J. M. and Calvo M. P. (1994) *Numerical Hamiltonian Problems.* Chapman and Hall, London.

[SW88] Sussman G. J. and Wisdom J. (1988) Numerical evidence that the motion of pluto is chaotic. *Science* 241: 433–437.

[Tan87] Tanekusa J. (1987) Statistical theory of violent relaxation. *Publications of the Astronomical Society of Japan* 39: 425–436.

[TG00] Tsuchiya T. and Gouda N. (January 2000) Relaxation and lyapunov time scales in a one-dimensional gravitating sheet system. *Physical Review E* 61: 948–951.

[TGK96] Tsuchiya T., Gouda N., and Konishi T. (March 1996) Relaxation processes in one-dimensional self-gravitating many-body systems. *Physical Review E* 53: 2210–2216.

[Vil82] Villumsen J. V. (May 1982) Simulations of galaxy mergers. *Monthly Notices of Royal Astronomical Society* 199: 493–516.

[WGL+98] Warren M. S., Germann T. C., Lomdahl P. S., Beazley D. M., and Salmon J. K. (1998) An alpha/linux cluster achieves 10 gflops for $150k. In *Proceedings of SC98*, pages (CD–ROM). ACM.

[WH91] Wisdom J. and Holman M. (1991) Symplectic maps for the n-body problem. *The Astronomical Journal* 102: 1528–1538.

[WHG96] White C. A. and Head-Gordon M. (1996) Rotating around the quartic angular momentum barrier in fast multipole method calculations. *J. Chem. Phys.* 105: 22–28.

[WOT+95] Woo S. C., Ohara M., Torrie E., Singh J. P., and Gupta A. (1995) The splash-2 programs: Characterization and methodological considerations. In *Proceedings of the 22nd International Symposium on Computer Architecture*, pages 24–36. ACM.

[WS92] Warren M. S. and Salmon J. K. (1992) Astrophysical N-body simulations using hierarchical tree data structures. In *Supercomputing '92*, pages 570–576. IEEE Comp. Soc., Los Alamitos.

[Yos95] 吉田春夫. (1995) シンプレクティック数値解法. 数理科学 384:.