疑似粒子多重極法とその展望

牧野淳一郎、川井敦 (東大総合文化)

要約

新しい高次精度の高速多重極展開法とツリーコードの実現方法を定式化した。これは疑似粒子で多重極展開を表現する。特徴は、

- 従来の方法よりもずっと実装が簡単であり
- GRAPE を有効に使うことが出来る

ことである。

話す内容

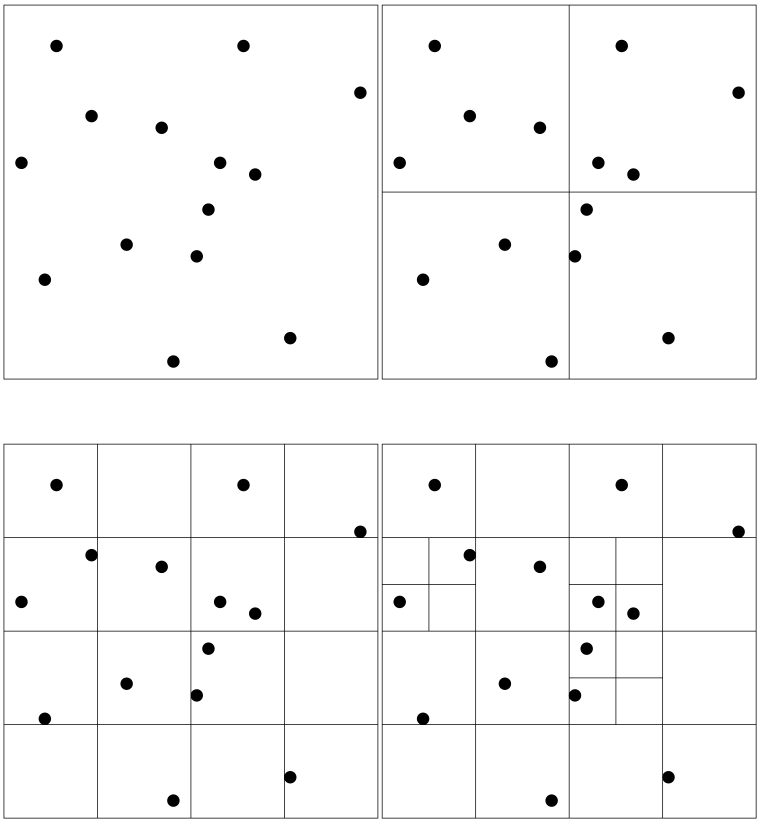
1. 高速多重極展開法とツリーコードの原理
2. Pseudoparticle method の考え方
3. 数値例
4. まとめ

高速多重極展開法とツリーコードの原理

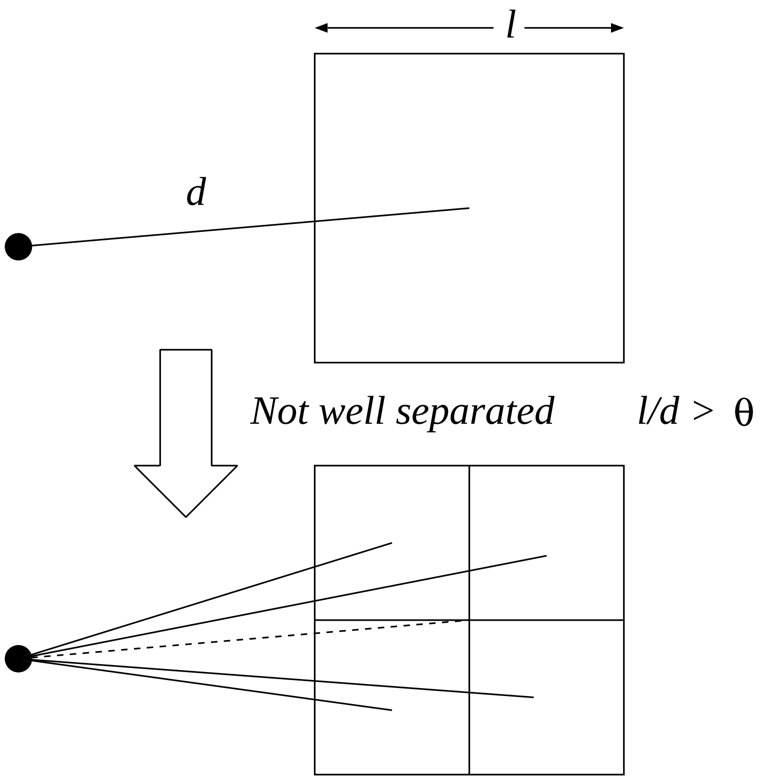
- どちらも、基本的には遠くの粒子をグループにまとめることで計算量を減らす。

ツリーコードの原理：

- 遠くの粒子からの力はまとめる。
- まとめるのに、階層的なツリー構造を使う。



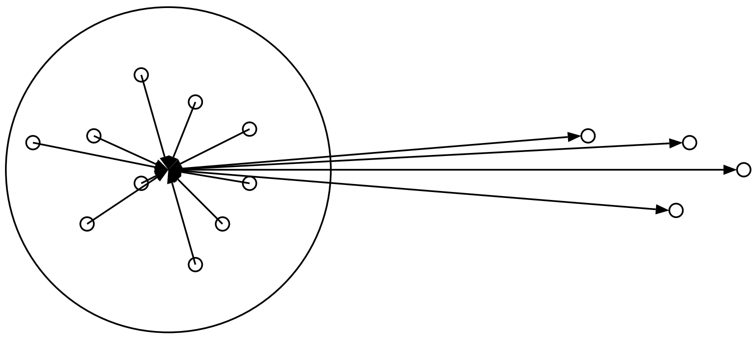
ツリーコードでの力の計算



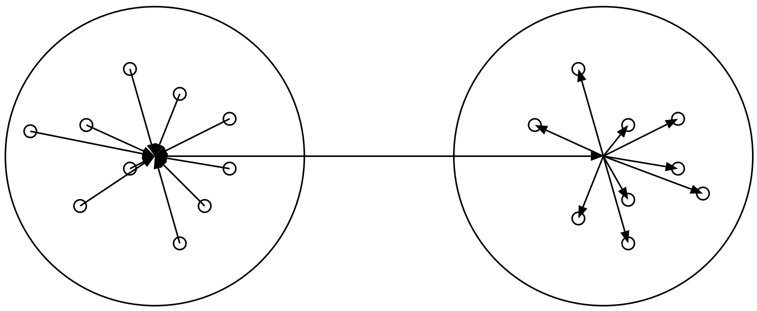
- 十分に離れている：重心（あるいは多重極展開）からの力
- そうでない：子ノードからの力の合計

系全体からの力 ＝ ルートノードからの力
あとは再帰的に計算される。

高速多重極展開法



Tree



FMM

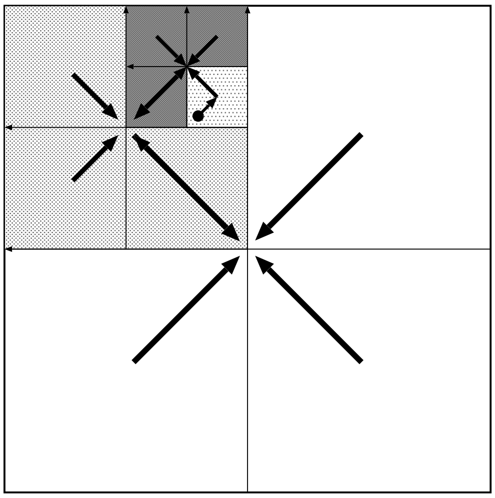
- ツリー：力を及ぼすほうだけをまとめて評価
- FMM：力を受けるほうもまとめて評価

力を受ける方の計算法：

1. 規準点でのポテンシャルの調和関数展開を計算する。これを「局所展開」と呼ぶ。
2. 各粒子の位置での力を展開係数から評価する。

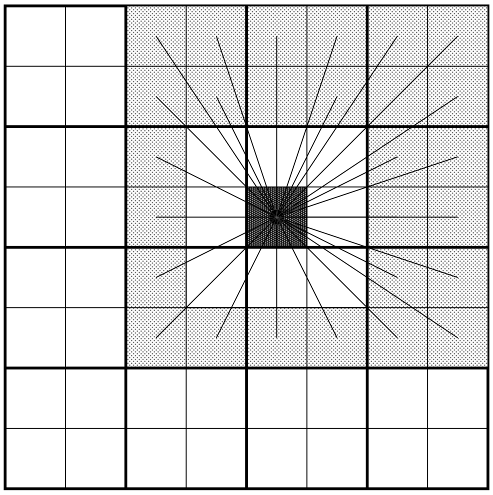
実際の計算法：

- 各セルが外に作るポテンシャルの球面調和関数展開（多重極展開）を、自分の下のセルの多重極展開の展開中心をシフトして求める。（upward path）

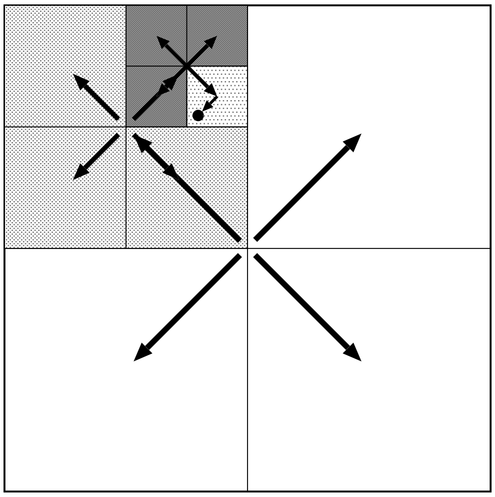


- 各セルが、
  - 自分と同じレベルで
  - 親の隣接セルの子で
  - 自分の隣接セルでない

  セルによるポテンシャルの和を自分の中心での調和関数展開にしておく（局所展開）



- 一番上のレベルから順に、自分の受けているポテンシャルを子セルに伝える（局所展開の中心をシフトする, downward path））



- 最終的には各粒子が自分の入っているセルの中心での局所展開を評価する。

多重極展開の算数：3 次元の場合

展開係数

$$\alpha_l^m = \sum_{i=1}^{N} m_i r_i^l Y_l^{-m}(\theta_i, \phi_i)$$

球面調和関数

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

多重極展開

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

局所展開

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

計算コードの実現には、

- α_k から β_k への変換
- α_k の展開中心のシフト
- β_k の展開中心のシフト

が必要：どれも非常に面倒

多重極展開の算数： 3 次元の変換式

$$\mathbf{M}_{n'}^{m'} = \sum_{n,m} \mathbf{T}^{M2M}_{n,m,n',m'}(\vec{\rho})\mathbf{M}_n^m$$

$$\mathbf{T}^{M2M}_{n,m,n',m'}(\vec{\rho}) = \begin{cases} \frac{(-1)^{n'-n} A_n^m A_{n'-n}^{m'-m}}{A_{n'}^{m'}} Y_{n'-n}^{*m'-m}(\alpha,\beta)\rho^{n'-n}, & \text{if } n' \geq n \text{ and} \\ & \text{if } |m'-m| \leq n'-n \\ 0, & \text{otherwise} \end{cases}$$

$$A_n^m = \frac{1}{\sqrt{(n+m)!(n-m)!}}$$

$$\mathbf{L}_{n'}^{m'} = \sum_{n,m} \mathbf{T}^{M2L}_{n,m,n',m'}(\vec{\rho})\mathbf{M}_n^m$$

$$\mathbf{T}^{M2L}_{n,m,n',m'}(\vec{\rho}) = \frac{(-1)^{n'+m} A_n^m A_{n'}^{m'}}{A_{n'+n}^{m'-m}} \frac{Y_{n'+n}^{*m'-m}(\alpha,\beta)}{\rho^{n'+n+1}}$$

$$\mathbf{L}_{n'}^{m'} = \sum_{n,m} \mathbf{T}^{L2L}_{n,m,n',m'}(\vec{\rho})\mathbf{L}_n^m$$

$$\mathbf{T}^{L2L}_{n,m,n',m'}(\vec{\rho}) = \begin{cases} \frac{A_{n'}^{m'} A_{n-n'}^{m-m'}}{A_n^m} Y_{n-n'}^{m-m'}(\alpha,\beta)\rho^{n-n'}, & \text{if } n \geq n' \text{ and} \\ & \text{if } |m-m'| \leq n-n' \\ 0, & \text{otherwise} \end{cases}$$

(From Elliot and Board, 1994)

- 面倒である。
- 高速計算が困難 — Y_n^m が漸化式で表されるので並列性が低い

Pseudoparticle method

FMM, ツリーを高精度にする上の問題

- プログラムを書くのが大変
- GRAPE が使えない = あまり速くない

- FMM や高次の展開を使ったツリーコードを簡単に実装でき
- GRAPE を有効に使える

ような方法はないか？

計算量がもっとも多いところ：多重極展開から局所展開への変換（FMM）、または多重極展開の評価（ツリーコード）

↓

多重極展開が「疑似粒子の集まり」で表現されていれば、どちらも疑似粒子の作るポテンシャルを評価するだけですむ。

↓

GRAPE を使って計算できる *!?*

多重極：粒子（電荷）の配置の極限
→「うまく」粒子を置くのは当然出来るはず。

問題：
いくつ粒子がいるか？
どうやって配置を決めるか？

粒子配置の考え方

- 自由度の数から考える

  粒子 **1** つの自由度： **4** 個

  p 次までの球面調和関数の全自由度： $(p+1)^2$

  $\rightarrow$ 例えば **3** 個の粒子で $p=2$ を表現出来る???

  - 非線形の連立方程式を解かないといけない
  - 有意味な解があるかどうか良くわからない

- 粒子の空間配置は固定して、質量だけ変える

  - （欠点）必要とする粒子の数が増える
  - （利点）線形計算で配置を決めることが出来る
  - （利点）いつでも配置を求められる

とりあえず粒子を置く座標はあらかじめ決めておくやりかたについて考える。

粒子配置の原理： 2 次元の場合

適当な半径の円周上に等間隔で疑似粒子を配置

↓

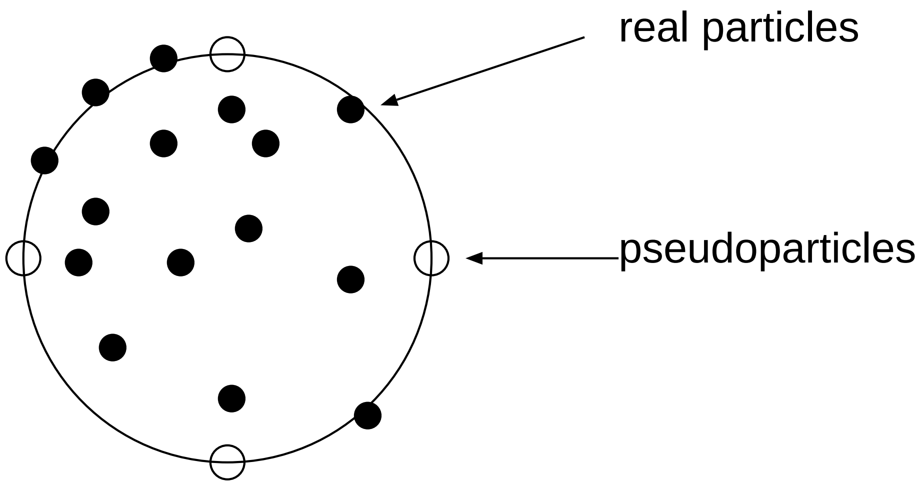
疑似粒子がつくるポテンシャルの多重極展開は、結局質量分布を角度方向でフーリエ級数展開したもの

↓

多重極展開の係数を逆フーリエ変換すれば、質量分布を決められる。

もとの粒子分布からいきなり求めると、

$$\begin{aligned} m_j &= \frac{1}{2p+1}\sum_{k=0}^{p}\sum_{i=1}^{n} m_i (z_i/z_j)^k \\ &= \frac{1}{2p+1}\sum_{i=1}^{n} m_i \frac{1-(z_i/z_j)^{p+1}}{1-z_i/z_j} \end{aligned}$$

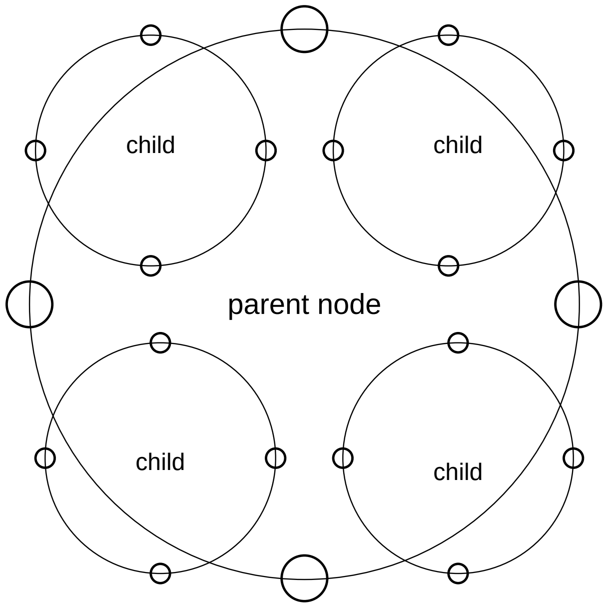


$2p+1$ 個の粒子で、 p 次までの展開を完全に表現できる。

多重極展開のシフト： 2 次元の場合

ある点の回りの疑似粒子分布のシフトにも、粒子を置き換える式がそのまま使える。

下のレベルの疑似粒子を物理粒子とみなすだけ。



粒子の配置の数学： 3 次元の場合

3 次元では、粒子を球面上におく。配置が **“spherical t-design”** となるようにとる必要がある。（球面調和関数の直交性を実現するため）

球面上の点が **spherical t-design** をなす
$=$ t 次までの任意の多項式の球面上での積分が、それらの点でのその多項式の値の平均に等しい。

例：正 4 面体の頂点： **1-design**, 正 20 面体の頂点： **4-design**。**20-design**（**216**点）といったものも作られている。正多面体は 20 以上はもうないので、準正多面体とかいろいろ使う。

あとは 2 次元の場合と同様。

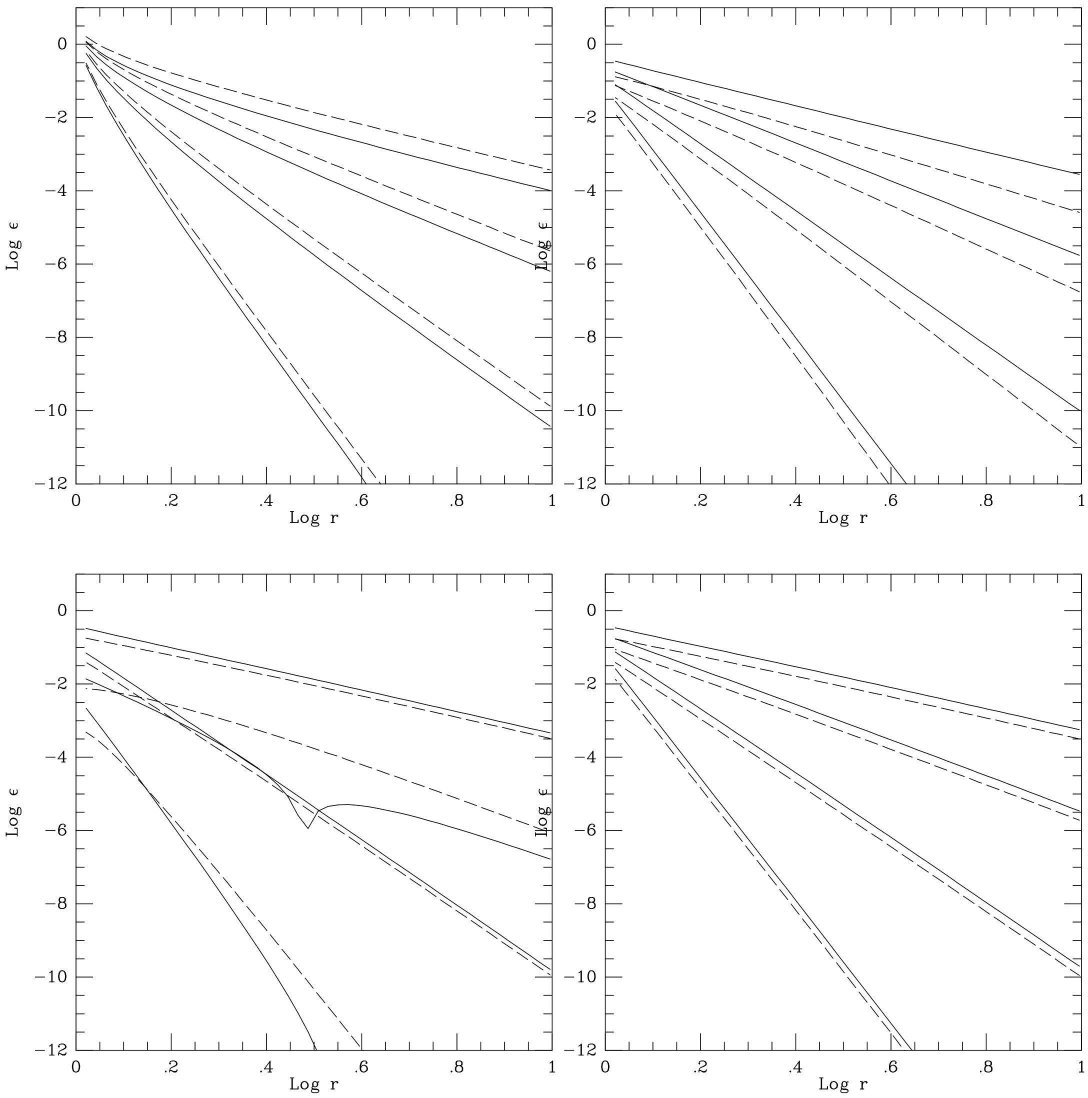
粒子分布から疑似粒子の質量を決める式:

$$m_j = \frac{2l+1}{K} \sum_{i=1}^{N} m_i \sum_{l=0}^{p} (r_i/r)^l P_l(\cos\gamma)$$

ここで γ は粒子の位置と疑似粒子の位置の間の角度。

数値例

2 次元で、粒子 1 個の作るポテンシャルの誤差を評価してみた。



粒子の位置： (1,0), 疑似粒子をおく円の半径： 0.75

ポテンシャルを評価する位置の方向： 0 (左上), $2\pi/3$ (右上), $\pi/2$ (左下), π (右下),

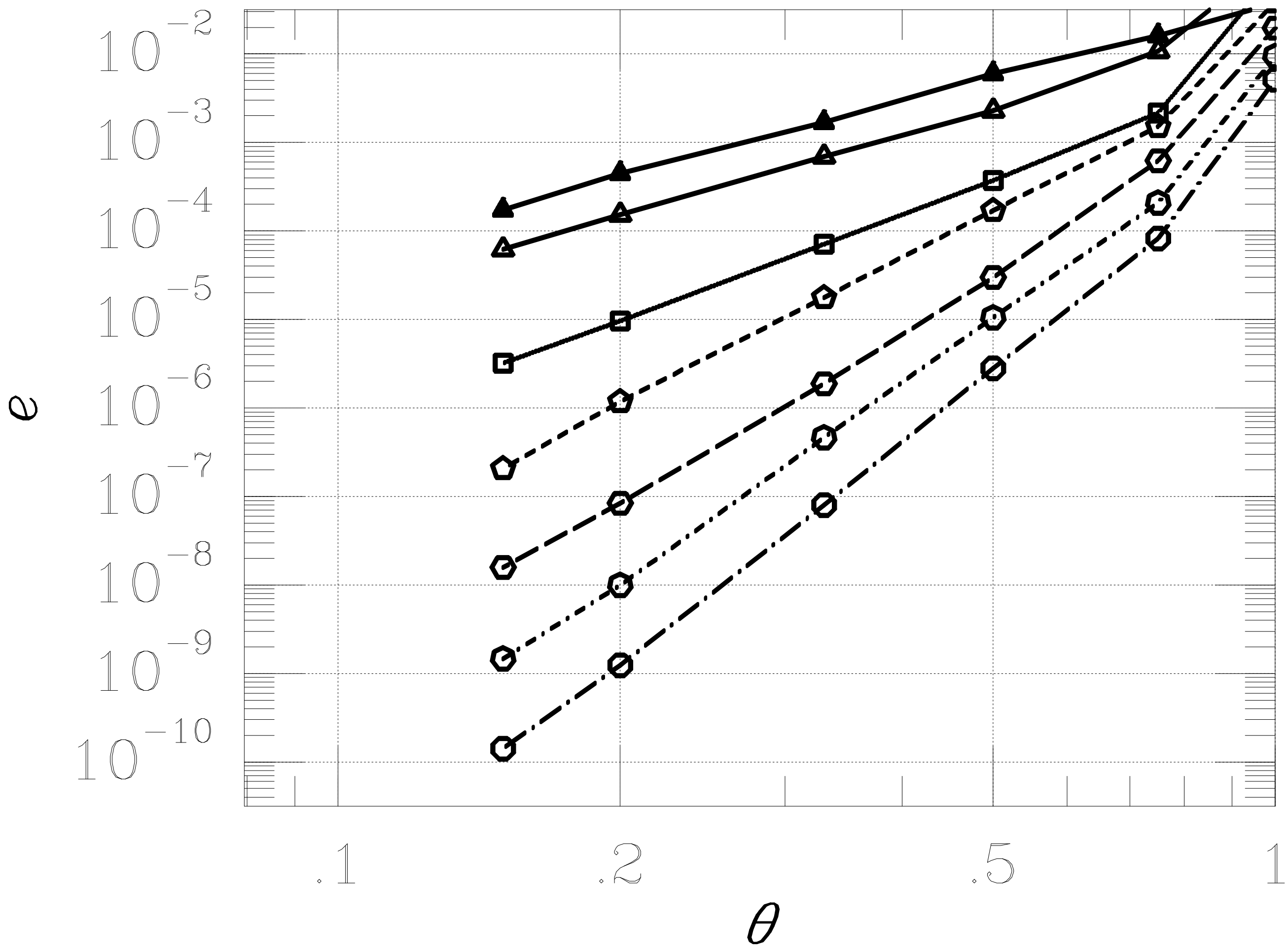
展開次数： 2, 4, 8, 16

実線：疑似粒子, 破線：調和関数展開

同等の精度が得られた

ツリーコードへの実装例 — 見込み角と計算精度

以下の計算結果は川井による。



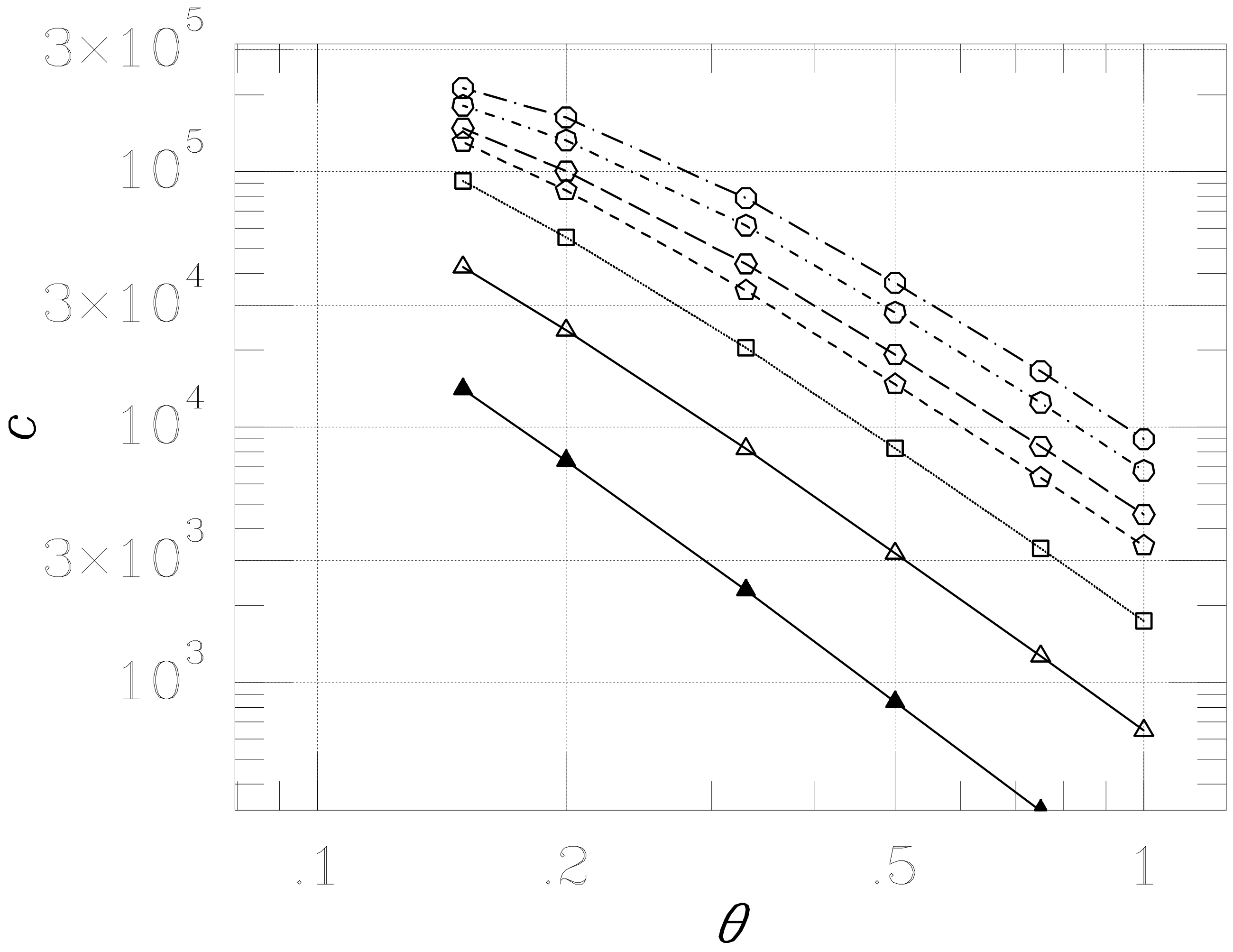
粒子数 **26** 万体、一様球

上から普通のツリーコード、 $p = 1 ... p = 6$

書いているものは加速度の相対誤差の平均値

確かに高次にすれば高い精度が達成できる

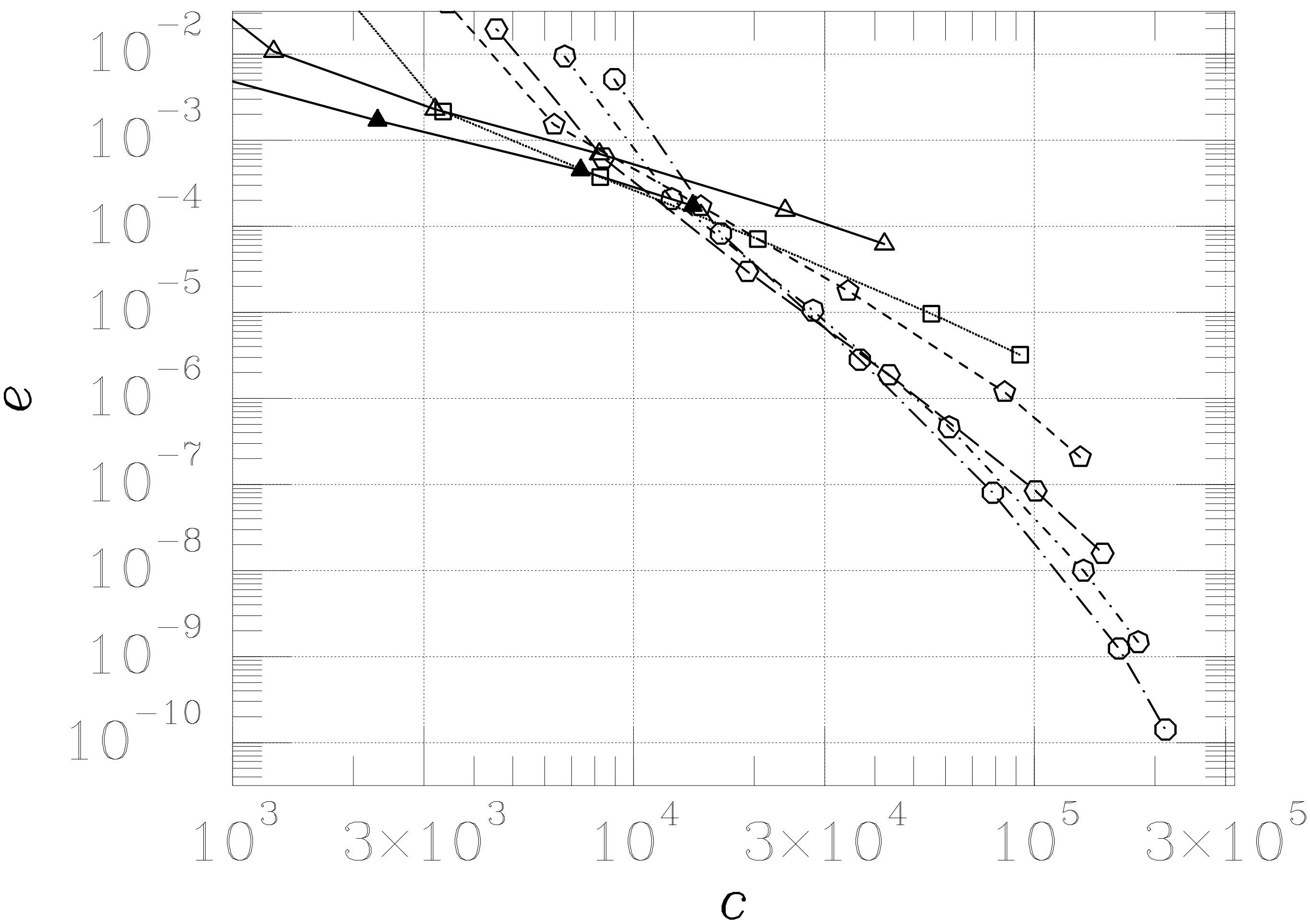
見込み角と計算量



縦軸 c は一粒子当たりの計算量（相互作用する粒子の数）

ほぼ p^2 で計算量が増える

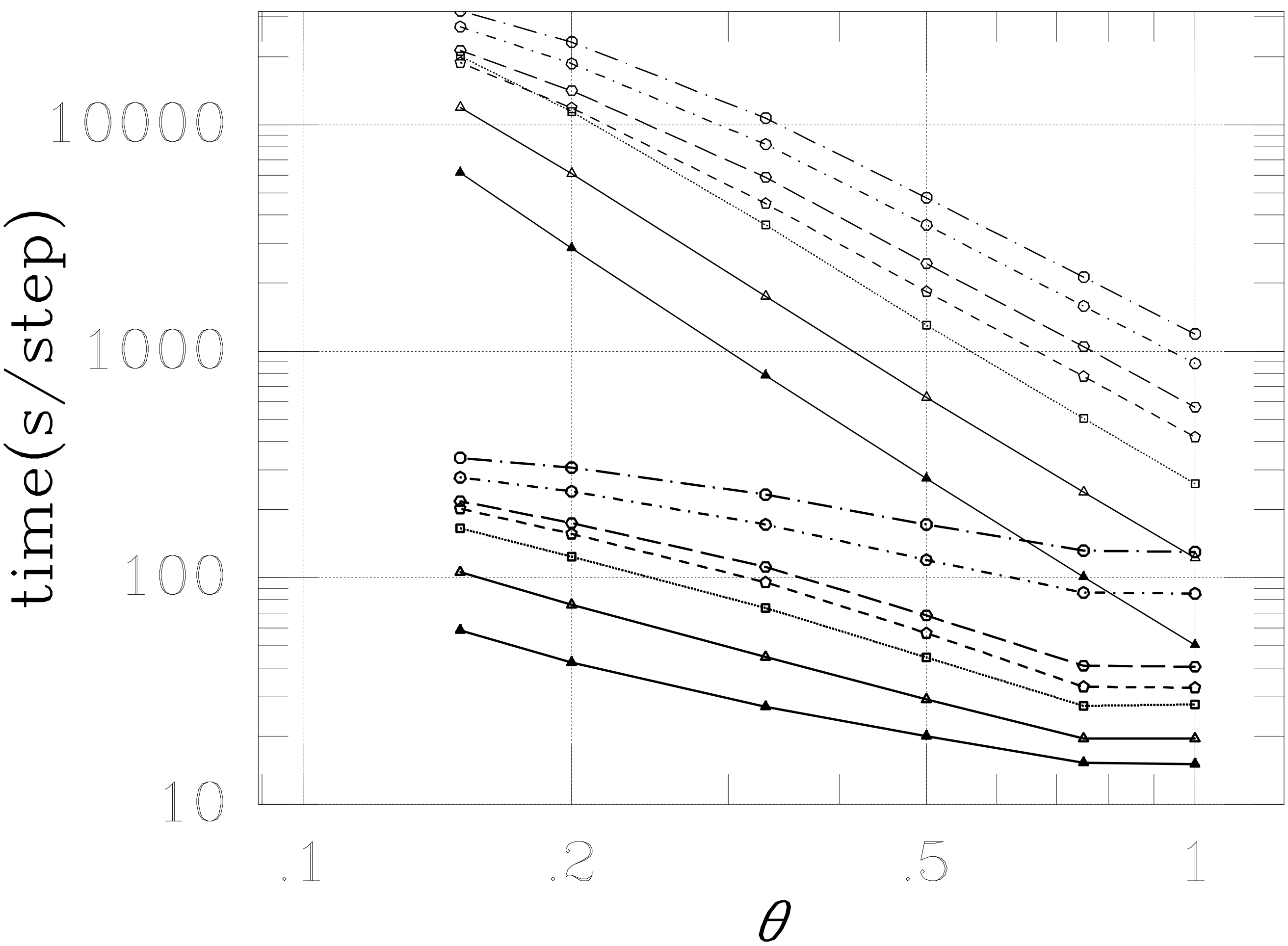
計算量と計算精度



高精度を要求すれば、高次が有利

本当は粒子分布によるはず

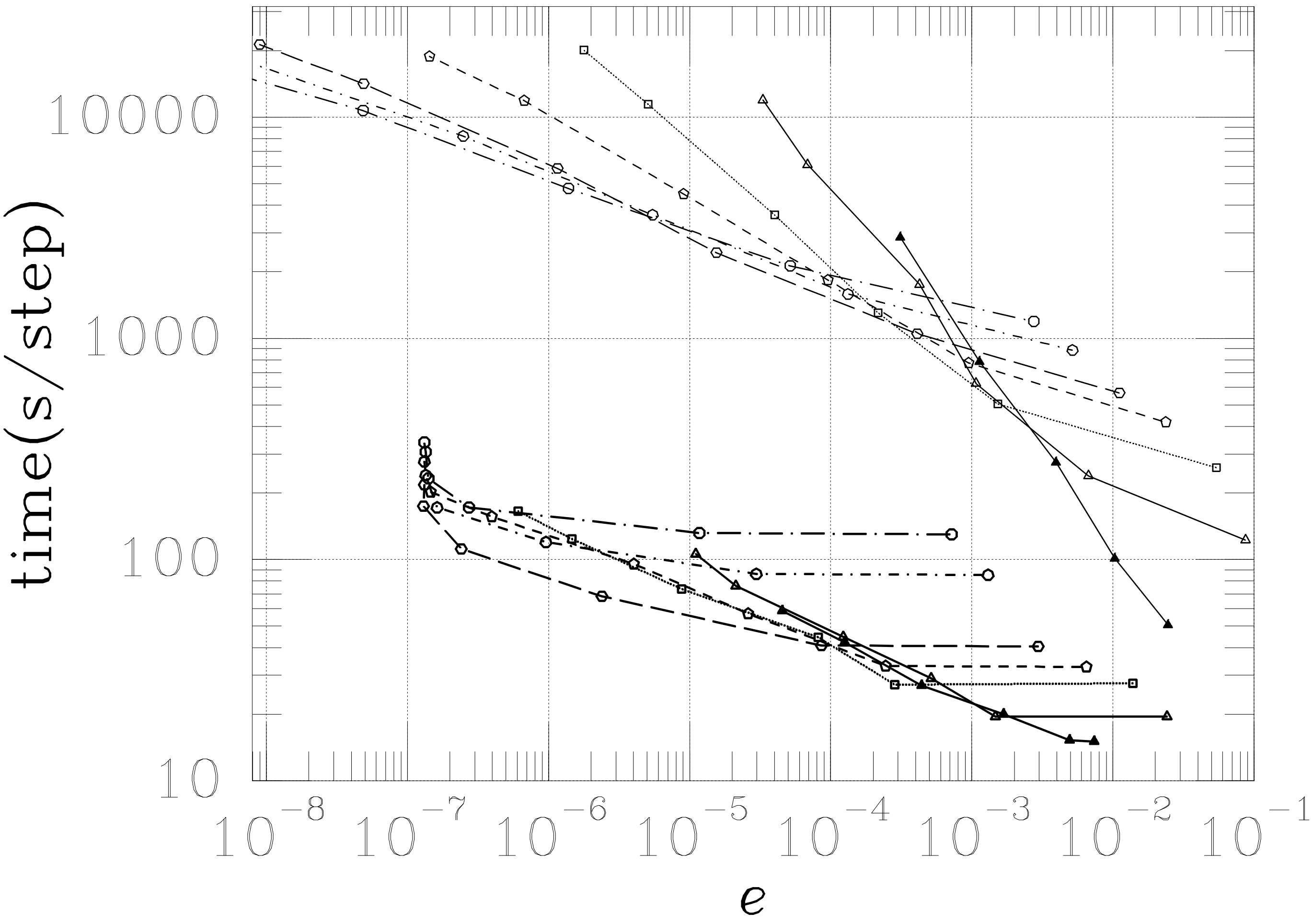
見込み角と計算時間



上はワークステーション (**DEC Alpha EV5 300 MHz**)
下は **GRAPE-4** 一枚 (**15 Gflops**)

GRAPE では見込み角を小さくしても計算量があまり増えない (あらかじめ多い)

計算精度と計算時間



上はワークステーション (DEC Alpha EV5 300 MHz)
下は GRAPE-4 一枚 (15 Gflops)

GRAPE ではハードウェアの精度限界が見える

まとめ

- 高速多重極展開法にもツリーコードにも使える、多重極展開を粒子分布で表現する方法を 2 次元と 3 次元の両方で定式化した。
- 2 次元の場合には、簡単な数値例で定式化が正しいことと精度が従来の方法にくらべてそれほど悪くないことを確認した
- この方法の実装は、従来の方法よりもはるかに容易である。
- これにより、世界で初めて任意次数のツリーコードを実現した。
- この方法では、高精度の場合でも計算量のもっとも大きいところを GRAPE で実行でき、ツリーコードの場合 大幅な高速化が実現できた。

検討課題

- 独立時間刻みへの対応
  - 球状星団、惑星形成への適用
- 実用的な、低次の場合の疑似最適化
  - 1 次で 4 点、 2 次で 12 点も使うのはちょっとおまぬけ