2004－EVA－8　(6)
2004／3／18

汎用ハミルトン系シミュレーションシステムとしてのシンプレクティック・レイトレーシング：並列処理系への実装と計算コスト評価

佐藤 哲

独立行政法人 通信総合研究所

シンプレクティック・レイトレーシングは，光線が複雑に曲進する状況下でコンピュータ・グラフィックスを生成する手法である．しかしながら画像を生成する手法にも関わらず，処理時間の大部分は，描画ではなく力学系のシミュレーションが閉めており，処理の高速化のためには数値シミュレーションの効率化が必要となる．本報告では，自動微分と数値積分を用いて汎用的なハミルトン系のシミュレーションが可能なシステムの概要を説明し，実装結果を処理の高速化の観点から検討する．

Symplectic ray tracing for simulation of a Hamiltonian system: implementation in parallel computers and evaluation of the calculation cost

Tetsu R. Satoh
Communications Research Laboratory

This paper reports an outline of the mechanism of the general-purpose simulation which is based on an automatic differentiation and symplectic integration. A symplectic ray tracing is a method of generating computer graphics by considering curved light rays. Though the method is used for rendering computer graphics, most parts of the process are occupied by not rendering but numerical simulation. Therefore, developing an algorithm of effective numerical simulation is required for fast processing. In this paper, implementation results and evaluation of the calculation cost from the viewpoint of high-speed computation are described.

1　はじめに

物理現象の可視化のために提案されたシンプレクティック・レイトレーシングは，描画のための前処理として数値シミュレーションが実行されるため，力学系のシミュレーションシステムとして用いることができる．シミュレーションは，ハミルトニアンを関数として与えることで実行することができ，自動微分法により自動的にハミルトニアンから正準方程式を導出し，シンプレクティック積分法により数値解を求める一連の動作がシステムに含まれる．また，スカラー関数からシミュレーションされるように設計されているので，力学系が定義される空間あるいは多様体の次元を問わずに適用可能である．現在のところ問題となるのは，コンピュータ・グラフィックス生成手法として用いるためには計算時間が長いことがあげられる．そこで本論文では，主に計算速度向上の観点からシステムの処理を考察し，今後の展望について述べる．

2　可視化技術としてのシンプレクティック・レイトレーシング

本節では，コンピュータ・グラフィックス（CG）生成のために数値シミュレーションが必要となる背景について述べる．

シンプレクティック・レイトレーシングは非線形光線追跡法 [1] と呼ばれる CG 描画手法の一つである．視線と配置オブジェクトの交点を検出し，交点の配置オブジェクトの色を描画することで CG を描画するという光線追跡法 [2] の概念に基づき，CG を作成する．図 1 は右端に観測者，左端に宇宙の銀河が存在することを表しており，中央が作成される CG である．観測者と銀河の間にブラックホールのような重い天体が存在すれば，銀河から出た光は点線のような軌道で観測者の視界に入る．しかし観測者は光

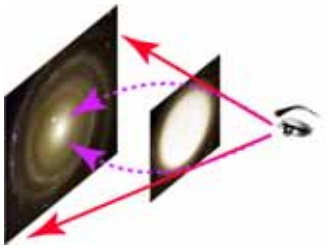


図 1: 曲がる光線により歪んだ映像が観測されることの概念図

線は図中の実線のように直進してきたと認知するので，結果的に中央のように銀河中心部の白い部分が大きく引き伸ばされた歪んだ光景を観測してしまう．このような歪みにより，間接的に非線形光線現象を可視化する．この時，曲がる光線の軌道を計算することは，光線の軌道を表す微分方程式の初期値問題に帰着するため，数値計算が必要となる．本システムでは，光線の軌道を表す方程式としてハミルトンの正準方程式 [6]

$$\begin{cases} \dfrac{dp}{dt} &= -\dfrac{\partial H}{\partial q} \\ \dfrac{dq}{dt} &= \dfrac{\partial H}{\partial p} \end{cases} \tag{1}$$

を採用する．ここで，q は位置座標成分，p は q に対応した運動量である．H はハミルトニアンと呼ばれるスカラー関数である．

光線が曲がる物理現象としては，地球上では密度が不均質な媒体による光線の屈折現象，宇宙では重い天体による光線の湾曲現象などがあげられる．通常，前者に対しては屈折率関数を含む微分方程式が，後者に対しては微分幾何学における測地線の微分方程式が用いられる．本研究であえてそれらの方程式を採用せずに正準方程式を採用したのは，以下のような目的があったからである．

1. 実装の手間と理論的な解析のし易さの両方の面を考え，扱う方程式を一つに統一したい
2. 保存量を持つ微分方程式の数値解法を使い，数値計算の誤差を推測したい

これらの目的は，正準方程式及び次節で述べる自動微分法，シンプレクティック積分法を採用することで達成できる．

3 自動微分と数値積分

3.1 高速自動微分法

前節において，式（1）を用いれば多くの現象を統一的に扱えることを紹介したが，プログラムとして実装するためにはハミルトニアンを準備し，ハミルトニアンを式（1）に従って偏微分した方程式を書き下し，サブルーチンなどの形でプログラミングしなければならない．しかしそれでは理論的には形式的に統一的に扱えても，実装としては別々に扱わなければならず，真に統一的に扱えるとは言えない．そこで，偏微分して方程式を書き下すという部分を自動化するための技術が自動微分法 [7][8] である．自動微分法では，ある演算を実行する際に演算の微分も同時に計算し，レジスターに蓄積していく．例えば，乗算という演算を自動微分法として実装する場合，次のように乗算の計算と同時に乗算の微分であるライプニッツ則を計算する．C++言語では，例えば次のような実装になる．

```
autodiff autodiff::operator*(autodiff a)
{
  autodiff ret;
  ret.value = value * a.value;
  ret.diff = value*a.diff + diff*a.value;
  return(ret);
}
```

このような演算を再帰的に実行することで，出力レジスタには演算結果と共に導関数の値が格納される．乗算の実装例からも分かるように，微分の計算規則をプログラム中に内蔵させるので微分を正確に計算することになり，数値微分のような打切り誤差は発生しない．

自動微分法を導入することで，式（1）を採用して非線形光線追跡法を実行するためには，ハミルトニアンの値を計算するサブルーチンを実装すれば，ハミルトニアンの導関数の値も自動的に計算され，式（1）の方程式が自動的に導かれる．従って自動微分法により局所的な光線の軌道を計算する方程式が自動的に導出されることになる．

3.2 シンプレクティック数値解法

自動導出された方程式は，解を求めなければならない．そのために数値積分法を用いれば，数値計算の誤差の範囲で微分方程式の解を人手による式変形を必要とせずに解を求めることができる．オイラー法や Runge-Kutta 法といった有名な数値積分法は，計算結果に打切り誤差を含むために誤差の蓄積に注意しなければならない．ここで，非線形光線追跡法においては，オブジェクトと光線の交差点を正確に求めることが大切であり，その過程の光線の軌道が正確かどうかは重要でないという点に着目する．表現を変えると，局所的な数値誤差よりも大域的な数値誤差を小さくすれば良いと言える．このような目的に対して開発された数値解析法の一つに，幾何的数値解析法 [9] がある．幾何的数値解析法の中で，ハミルトンの正準方程式を扱う場合に適するシンプレクティック数値積分法 [10] [11] がある．シンプレクティック数値積分法は，正確には次のように定義されるシンプレクティック性を保存量として持つ．

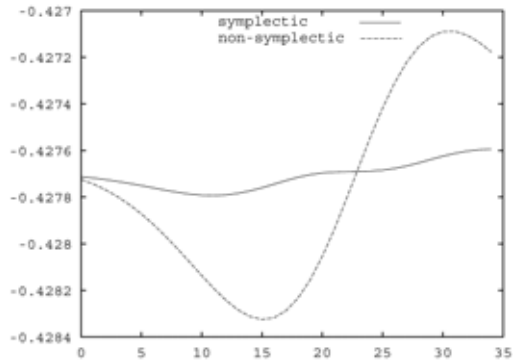


図 2: 蚤気楼が発生している状況で光線追跡を行った場合のエネルギー値

定義 1 $(p(t_0), q(t_0))$ を $t = t_0$ での光子の座標であるとする．$(p(t_0 + \Delta t), q(t_0 + \Delta t))$ は $t = t_0 + \Delta t$ での座標であるとする．任意の実数 t_0 に対し，写像 $(p(t_0), q(t_0)) \mapsto (p(t_0 + \Delta t), q(t_0 + \Delta t))$ がシンプレクティック性を保存するのは，

$$dq(t_0) \wedge dp(t_0) = dq(t_0 + \Delta t) \wedge dp(t_0 + \Delta t) \quad (2)$$

が成り立つ時およびその時に限る．ここで，dp , dq は各変数の *1*-形式であり，$\wedge$ は微分形式の外積を表している．

この手法は長期間の数値積分に強い特徴があり，安定性が高い．図 2 に，非線形光線追跡法を用いて蚤気楼のシミュレーションを実行した場合の，光線の軌道上でのエネルギーの値を示す．縦軸がエネルギーの値で，横軸が光線を発射してからの光線の経路長を表すパラメータである．破線が Runge-Kutta 法を用いて正準方程式を解いたものであり，実線がシンプレクティック法を用いたものである．打切り誤差の精度はどちらも 4 次である．数値計算の刻み幅も打切り誤差の精度も同じであるにも関わらず，シンプレクティック法の方がエネルギーを良好に保存していることが分かる．ここではシンプレクティック法として，半陰的オイラー法

$$\begin{cases} p_{k+1} &= p_k - \tau \dfrac{\partial H(p_{k+1}, q_k)}{\partial q_k} \\ q_{k+1} &= q_k + \tau \dfrac{\partial H(p_{k+1}, q_k)}{\partial p_{k+1}} \end{cases} \quad (3)$$

に対し対称分解法 [12][13] を用いて精度を高めたものを使った．具体的には，光線の運動に対するハミルトニアンは多くの場合に運動量成分について分離可能となるので，それを H_A，H_B とおく：

$$H(p, q) = H_A(p, q) + H_B(p, q) \quad (4)$$

そして刻み幅 τ で式 (3) を用いて H_A，H_B をハミルトニアンとする正準方程式を解くルーチンを $S_A(\tau)$, $S_B(\tau)$ とすると，2 次の精度を持つ解法が次のように構成される：

$$S_2(\tau) \equiv S_A(1/2\tau) S_B(\tau) S_A(1/2\tau) \quad (5)$$

同様に，

$$S_4(\tau) \equiv S_2(d_1\tau) S_2(d_2\tau) S_2(d_1\tau) \quad (6)$$

が 4 次の精度を持つ解法となる．ここで，d_1，d_2 は

$$\begin{cases} d_1 &= \dfrac{4 + 2^{2/3} + 2^{4/3}}{6} = 1.35121\cdots \\ d_2 &= -\dfrac{(1 + 2^{1/3})^2}{3} = -1.70241\cdots \end{cases} \quad (7)$$

と定義される定数である．ただし式 (7) を用いると，結果的に数値計算の刻み幅 τ が d_1 倍，d_2 倍されることになるため，τ として大き目の値を使いたい場合は解法 (6) , (7) では精度や安定度が落ちる場合がある．その場合は例えば

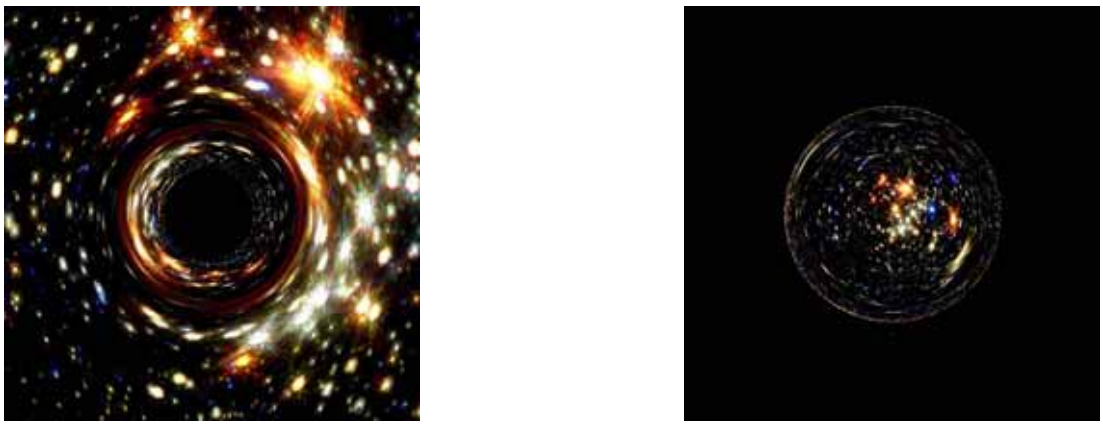
$$\begin{aligned} S_4(\tau) \equiv\ & S_2(d_1\tau) S_2(d_1\tau) S_2(d_1\tau) S_2(d_1\tau) S_2(d_2\tau) \\ & \times S_2(d_1\tau) S_2(d_1\tau) S_2(d_1\tau) S_2(d_1\tau) \end{aligned} \quad (8)$$

$$\begin{cases} d_1 &= \dfrac{1}{6} = 0.16666\cdots \\ d_2 &= -\dfrac{1}{3} = -0.33333\cdots \end{cases} \quad (9)$$

のようなスキームを構成する．ただし S_2 の評価回数が増えるので計算速度は低下する．また，陰的な解法なので処理中で連立方程式を解く必要がある．本研究では連立方程式の解法はニュートン法を用いている．

ハミルトニアンが運動量成分と座標成分に分離できない場合，解法 (3) を用いた対称分解法が使えないため，任意のハミルトニアンに対しシンプレクティック解法となる，例えば Gauss-Legendre 法

$$\begin{aligned} P_A &= p_k - \tau \left\{ \frac{1}{4} \frac{\partial H(P_A, Q_A)}{\partial Q_A} \right. \\ &\qquad \left. + (\frac{1}{4} - \frac{\sqrt{3}}{6}) \frac{\partial H(P_B, Q_B)}{\partial Q_B} \right\}, \\ Q_A &= q_k + \tau \left\{ \frac{1}{4} \frac{\partial H(P_A, Q_A)}{\partial P_A} \right. \\ &\qquad \left. + (\frac{1}{4} - \frac{\sqrt{3}}{6}) \frac{\partial H(P_B, Q_B)}{\partial P_B} \right\}, \\ P_B &= p_k - \tau \left\{ (\frac{1}{4} + \frac{\sqrt{3}}{6}) \frac{\partial H(P_A, Q_A)}{\partial Q_A} \right. \\ &\qquad \left. + \frac{1}{4} \frac{\partial H(P_B, Q_B)}{\partial Q_B} \right\}, \\ Q_B &= q_k + \tau \left\{ (\frac{1}{4} + \frac{\sqrt{3}}{6}) \frac{\partial H(P_A, Q_A)}{\partial P_A} \right. \end{aligned}$$



(a) (b)

図 3: (a) 球対称ブラックホールの可視化例．銀河の画像がブラックホールによる重力で歪んでいる．(b) 重力凹レンズ効果の可視化例．ブラックホールの近傍でブラックホールに背を向けると，このように宇宙全体が前方に集まって見える．

$$
\begin{aligned}
& + \frac{1}{4}\frac{\partial H(P_{\mathsf{B}}, Q_{\mathsf{B}})}{\partial P_{\mathsf{B}}} \Big\}, \\
p_{\mathsf{k}+1} &= p_{\mathsf{k}} - \frac{\tau}{2}\Big\{ \frac{\partial H(P_{\mathsf{A}}, Q_{\mathsf{A}})}{\partial Q_{\mathsf{A}}} \\
& + \frac{\partial H(P_{\mathsf{B}}, Q_{\mathsf{B}})}{\partial Q_{\mathsf{B}}}) \Big\}, \\
q_{\mathsf{k}+1} &= q_{\mathsf{k}} + \frac{\tau}{2}\Big\{ \frac{\partial H(P_{\mathsf{A}}, Q_{\mathsf{A}})}{\partial P_{\mathsf{A}}} \\
& + \frac{\partial H(P_{\mathsf{B}}, Q_{\mathsf{B}})}{\partial P_{\mathsf{B}}}) \Big\} \qquad (10)
\end{aligned}
$$

が使える．打切り誤差の精度は 4 次である．

以上述べたように，自動微分法と数値積分法を用いることにより，スカラー関数としてハミルトニアンの計算ルーチンを与えることにより光線追跡が可能となるシステムが実現する．ハミルトン力学で扱えるいかなる現象もハミルトニアンを与えるだけで実行できるため，多数の現象を統一的に扱えるようになると言える．

4 シンプレクティック・レイトレーシングの実装

4.1 画像生成例

シンプレクティック・レイトレーシング [14] [15][16] とは，非線形光線追跡法に対し自動微分法とシンプレクティック積分法を導入したものである．本研究では，この手法を c++を用いて実装しており，Linux PC，NEC SX-6，SGI Onyx と多くのプラットフォーム上で動作している．以下，実行例を紹介する．

図 3 は球対称ブラックホール時空内で光線追跡をした例で，ハミルトニアンは次のように表される：

$$
\begin{aligned}
H &= -\frac{1}{2}\left(1 + \frac{r_{\mathsf{g}}}{r}\right) p_{\mathsf{t}}^2 + \frac{1}{2}\left\{ \left(1 - \frac{r_{\mathsf{g}} x^2}{r^3}\right) p_{\mathsf{x}}^2 \right. \\
& \left. + \left(1 - \frac{r_{\mathsf{g}} y^2}{r^3}\right) p_{\mathsf{y}}^2 + \left(1 - \frac{r_{\mathsf{g}} z^2}{r^3}\right) p_{\mathsf{z}}^2 \right\} \\
& - \frac{r_{\mathsf{g}} p_{\mathsf{t}}}{r^2}\left(x p_{\mathsf{x}} + y p_{\mathsf{y}} + z p_{\mathsf{z}}\right) \\
& - \frac{r_{\mathsf{g}}}{r^3}\left(xy p_{\mathsf{x}} p_{\mathsf{y}} + xz p_{\mathsf{x}} p_{\mathsf{z}} + yz p_{\mathsf{y}} p_{\mathsf{z}}\right) \qquad (11)
\end{aligned}
$$

ここで，(t, x, y, z) は 4 次元カーテシアン座標系の成分，$(p_{\mathsf{t}}, p_{\mathsf{x}}, p_{\mathsf{y}}, p_{\mathsf{z}})$ は対応する運動量成分，$r = \sqrt{x^2 + y^2 + z^2}$，$r_{\mathsf{g}}$ はブラックホール半径と呼ばれるブラックホールの質量に対応する量である．図 3a と図 3b の違いは光線の発射方向，すなわち観測者の視線方向で，図 3a では観測者はブラックホールの方向を見ており，図 3b では観測者は逆にブラックホールに背を向けている．

図 4 は，公園の石畳が極端に熱された状況で光線追跡を実行した例である．温度の上昇に伴い地表面付近の空気の屈折率が変化し，逃げ水現象が発生している．この場合，ハミルトニアンは次のようになる．

$$
H = \frac{1}{2} n(x, y, z)(p_{\mathsf{x}}^2 + p_{\mathsf{y}}^2 + p_{\mathsf{z}}^2) \qquad (12)
$$

ここで $n(x, y, z)$ は，座標 (x, y, z) での屈折率を与える関数で，図 4 の作成時は

$$
n = 0.85 - 0.3z^3 + 0.03\cos(3\pi x) \qquad (13)
$$

と定義されている．なお，高さ方向 z 軸の原点はカメラの位置に設定されているので，地表付近では $z < 0$ となる．

4.2 計算時間についての考察

本節では，共有メモリ型並列計算機での計算時間を比較する．使用した計算機は，Intel Pentium III 800MHz のデュアル CPU マシン，Intel Xeon 2.8GHz のデュアル CPU マシン，MIPS R10000(250MHz) の SGI Onyx2，NEC SX-6 である．OS はいずれも

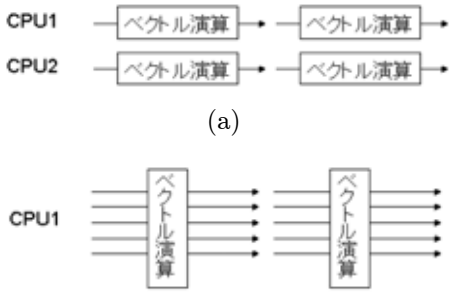
図 4: 蜃気楼現象としての逃げ水のシミュレーション例

UNIX 系で，コンパイラは Intel のプロセッサに対しては Intel C++，Onyx2 に対しては MIPSpro C++，SX-6 に対しては C++/SX を用いた．SX-6 では，横方向の光線の計算をベクトル化し，その計算を縦方向に繰り返すことで画像を生成した．画像の大きさは 50 画素 ×50 画素と 100 画素 ×100 画素の 2 通りを作成した．表 1 の中で，PC A は Pentium III プロセッサのマシンを示し，PC B は Xeon プロセッサのマシンを意味する．Onyx2 は古い計算機であるが，ほぼ CPU の個数に比例した計算速度が得られており，SGI の共有メモリアーキテクチャの良い性能によるものと思われる．SX-6 を使用した場合，他の計算機と比べ優れている結果は得られていないが，その原因は，この実装ではベクトル長が短すぎるからと思われる．表 1 を出力した実装では図 5a のように光線の一本一本をベクトル演算によって計算したが，図 5b のように複数の光線の計算をベクトル演算によって同時に実行する方が効率が良いと考えられる．簡単な実験で，画像の横一列をベクトル計算し，縦方向に並列処理をすると PC A よりは速い処理速度が得られた．横一列に留まらずより大量の画素の計算をベクトル化すると，より高速化可能と思われる．

また，一般に共有メモリ型計算機，分散メモリ型計算機のどちらにおいても，画素毎の負荷分散が大きな問題になる．現在のところ，生成画像中の各画素に対する光線追跡処理は，単純に左上から走査線状に下に向かって処理されているが，図 6 のように処理量は画素によって大きく異なり，順番に CPU を割り当てることは効率が悪い．図 6 は球対称ブラックホール時空で光線追跡をした場合，ブラックホールが存在する画面中央部では光線追跡期間が長くなり，数値誤差の蓄積が大きくなっていることを示している．このような場合，例えば横 1 ラインの単位

表 1: 計算時間測定結果（単位は分：秒）

$r_{\mathrm{g}} = 0.00, 50 \times 50$				
CPU	PC A	PC B	SX-6	Onyx2
1	1:24	0:38	1:51	13:28
2	0:45	0:33	2:21	7:39
$r_{\mathrm{g}} = 0.01, 50 \times 50$				
CPU	PC A	PC B	SX-6	Onyx2
1	5:42	2:08	13:42	57:01
2	3:05	1:28	5:58	28:19
$r_{\mathrm{g}} = 0.00, 100 \times 100$				
CPU	PC A	PC B	SX-6	Onyx2
1	5:41	2:42	8:14	
2	3:08	2:17	11:45	
$r_{\mathrm{g}} = 0.01, 100 \times 100$				
CPU	PC A	PC B	SX-6	Onyx2
1	22:51	8:48	34:38	
2	12:26	5:49	19:52	



(a)

(b)

図 5: ベクトル型計算機への実装方法

で並列処理をすると，両端は計算が終了していても中央部が終了せず，同期処理のために次のラインの計算が始まるまで両端の処理をした CPU にアイドル状態が発生することが考えられる．従って，前処理として全画面から適当な分布で疎に光線追跡をし，その結果より画面中の画素の計算に必要な負荷の分布を予測して CPU を割り振る処理を検討している．

4.3 数値積分法の種類の検討

既に述べたように，シンプレクティック積分法によりシミュレーションを実行する場合，計算速度や打切り誤差の精度が異なるいくつも解法の選択肢がある．本節では，対称分解法を用いる場合と Gauss-Legendre を対象に，ある精度を達成する場合の計算時間を見積もる．概算の方法としては，本システム

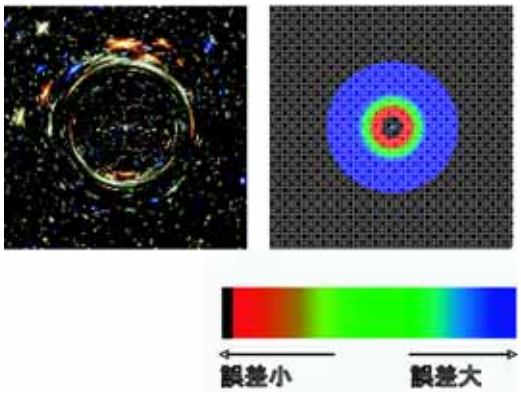


図 6: 誤差マップの作成

ではハミルトニアンの評価のコストがシステム全体の処理時間を支配していると考えられるので，ハミルトニアンの評価回数により見積もることにする．

まず対称分解法では，半陰的オイラー法（3）が実行される．これは一度にハミルトニアンが 2 回評価されるので，ニュートン法の反復回数を n 回とすると，評価回数は $2n$ 回である．そして式（5）を用いて精度を 2 次に上げる場合，$3 \times 2n = 6n$ 回の評価となる．そして式（6）を用いて精度を 4 次に上げると $3 \times 6n = 18n$ 回のハミルトニアン評価が必要となる．一方，Gauss-Legendre 法（10）では，ニュートン法の反復回数を n 回とすると，$8n + 4$ 回のハミルトニアン評価で 4 次の精度が達成できる．この二つの評価式の一次関数の傾きを考えると，一般に Gauss-Legendre 法の方が最終的な計算コストは低くなることが期待できる．ただし対称分解法の方が各連立方程式の次数が低くなるため，ニュートン法の反復回数は少なくなる可能性もある．

5 おわりに

本論分では，自動微分法とシンプレクティック積分法を用いて数値シミュレーションを行い，シミュレーション結果を用いて光線追跡法の概念に基づきコンピュータ・グラフィックスを生成する手法について述べた．現在のところ，適用対象が少ないこと，計算コストが高いことが問題として残されているので，適用できる現象や分野を検討し，また高精細な画像をリアルタイムに近い速度で生成することは可能かどうか検討する必要がある．

参考文献

[1] Gröller, E.: Nonlinear Ray Tracing: Visualizing Strange Worlds, *The Visual Computer*, Vol. 11, pp. 263–274 (1995).

[2] Whitted, T.: An Improved Illumination Model for Shaded Display, *Commun. ACM*, Vol. 23, No. 6, pp. 343–349 (1980).

[3] 斎藤泰, 牧野光則, 大石進一: レイトレーシング法を用いた異方性不均質透明体の表現, 信学論, Vol. J76-D-II, No. 8, pp. 1755–1762 (1993).

[4] Stam, J. and Languénou, E.: Ray Tracing in Non-Constant Media, *Proc. 7th Eurographics Workshop on Rendering*, pp. 225–234 (1996).

[5] 佐藤文隆, 小玉英雄: 一般相対性理論, 岩波書店, chapter 1: 多様体の力学, pp. 1–14 (1992).

[6] 原島鮮: 力学, 裳華房, chapter 14: ハミルトンの正準方程式, pp. 293–301 (1989).

[7] Iri, M., Tsuchiya, T. and Hoshi, M.: Automatic Computation of Partial Derivatives and Rounding Error Estimates with Applications to Large-Scale Systems of Nonlinear Equations, *J. Computational and Applied Mathematics*, Vol. 24, pp. 365–392 (1988).

[8] Griewank, A.: On Automatic Differentiation, *Mathematical Programming: Recent developments and Applications*, pp. 83–108 (1989).

[9] Hairer, E., Lubich, C. and Wanner, G.: *Geometric Numerical Integration–Structure-Preserving Algorithms for Ordinary Differential Equations*, Springer-Verlag Berlin Heidelberg (2002).

[10] Sanz-Serna, J.: Symplectic Intergrators for Hamiltonian Problems: An Overview, *Acta Numerica*, pp. 243–286 (1991).

[11] Yoshida, H.: Recent Progress in the Theory and Application of Symplectic Integrators, *Celestial Mechanics and Dynamical Astronomy*, Vol. 56, pp. 27–43 (1993).

[12] Suzuki, M.: Decomposition Formulas of Exponential Operators and Lie Exponentials with Some Applications to Quantum Mechanics and Statistical Physics, *J. Math. Phys.*, Vol. 26, No. 4, pp. 601–612 (1984).

[13] Yoshida, H.: Construction of Higher Order Symplectic Integrators, *Phys. Lett. A*, Vol. 150, pp. 262–268 (1990).

[14] 佐藤哲, 岩佐英彦, 竹村治雄, 横矢直和: シンプレクティック・レイトレーシング：ブラックホール 時空での光線追跡, 情処学論, Vol. 42, No. 3, pp. 456–464 (2001).

[15] 佐藤哲, 岩佐英彦, 竹村治雄, 横矢直和: 高速シンプレクティック・レイトレーシング：入れ子宇宙の 可視化, 情処学論, Vol. 42, No. 10, pp. 2392–2402 (2001).

[16] Satoh, T. R.: Symplectic Ray Tracing: A new approach to non-linear ray tracing by using Hamiltonian dynamics, *Proc. SPIE-IS&T Electronic Imaging, Visualization and Data Analysis 2003*, Vol. 5009, pp. 277–285 (2003).