MICS実験
並列数値シミュレーション

担当：仲谷

並列計算

一つの問題を複数の小さな問題に分割し、多くのプロセッサ（プロセッサーコア）で同時に計算

例：

- マルチコアCPU、GPU、PCクラスタ、超並列計算機等

- 利点
  - 高速化
    理想的には計算時間は1/プロセッサ数に減少
  - 大規模化
    問題の分割により、より大規模な計算が可能
- 欠点
  - 計算中のプロセス間通信のため、かえって遅くなる場合あり
  - 問題によっては並列に分割できない

課題及び進め方

- 非常に長い計算時間が必要になる銀河のシミュレーションを、GPU、並列計算機、PCクラスタ(MPI)のいずれかを用いて行う

課題	MPI/並列計算機	GPU
共通課題	○	○
MPI	○	
GPU		○

注：合格の必要条件 → 必修課題を全部解くこと

- 使用する計算機は、各自の希望を考慮し、実験の3日目に決定（目安：実験2日目の終了時までに、共通課題を終わっていないと、GPUのプログラムを書くのは難しい）

実験日程：

第3ラウンド

本日を含めて最初の3日間→最初に説明してから、各自で作業

残りの4日間　　→各自で作業（最初に出席をとる）

いずれの場合も、作業中はZoomにつないでおくこと

説明資料

- 今回及び今後の実験の説明資料は、以下のページからダウンロード可能

  注：学内からのみアクセス可

  http://wwwhnl.cs.uec.ac.jp/lecture/MICS/
  のページにある、資料(PDF)

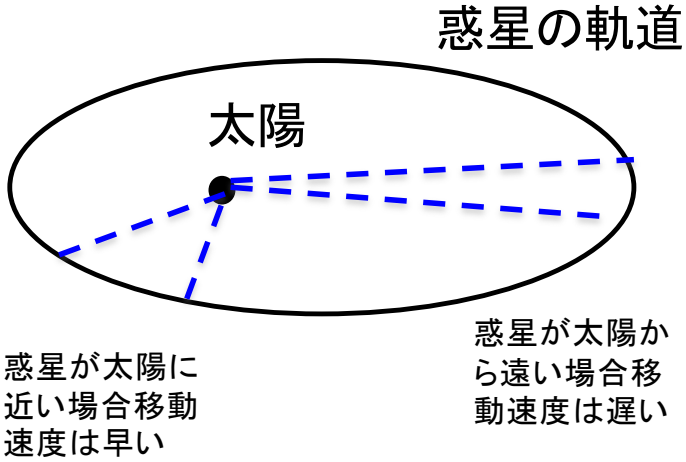
  ID,PASSWD: mics

  資料は次回から各自でダウンロードして用意すること

問題1：太陽の公転運動のシミュレーション

ケプラー（Kepler）の法則 (1619)

1. 惑星の軌道
   楕円軌道　(太陽は焦点の1つ)



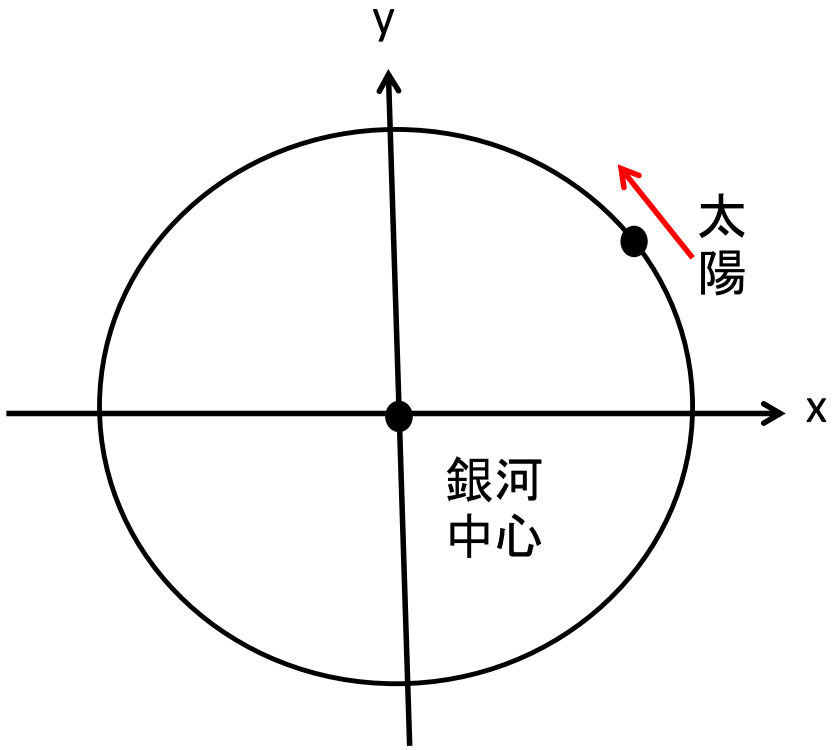
2. 太陽と惑星を結ぶ線分
   一定時間の移動面積は一定
   太陽と惑星との距離が変化
   →惑星の速さが変化

3. 全惑星の周期と半径の関係
   T^2/a^3 が一定
   (T:惑星の周期、a:楕円の半長軸)

   惑星の位置や速度の時間依存
   →*初等関数で表せない*
   →*シミュレーションによる数値解が必要*

（「太陽」を「銀河中心」、「惑星」を「太陽」に置き換えれば、太陽の公転運動でも上記が成り立つ）

数理モデル



- 太陽が一つ公転
  - 公転半径：3万光年
  - 公転周期：2億2600万年
- 銀河中心と太陽→質点
- 太陽軌道　　→円軌道
  →2次元平面内（x–y面内）
- 銀河中心：原点に静止
  →太陽の運動のみ計算

利用する法則

- 万有引力の法則
  - 太陽が銀河中心に引かれる力

$$\vec{F} = -\frac{GMm}{r^2}\frac{\vec{r}}{|r|} = -\frac{GMm}{r^3}\vec{r}$$

G:万有引力定数、M,m:銀河中心と太陽の質量、
r:銀河中心と太陽間の距離

- 運動の第二法則（ニュートンの運動方程式）
  - 物体に加わる力と加速度の関係

$$F = ma$$

F: 力、m:質量、a: 加速度

解くべき方程式

運動方程式

$$m\frac{d^2\vec{r}}{dt^2} = \vec{F} = -\frac{GMm}{r^3}\vec{r},$$

→二階微分方程式

計算機で解くために式変形

$$\frac{d^2x}{dt^2} = -\frac{GM}{r^3}x \qquad \Rightarrow \qquad \frac{dv_x}{dt} = -\frac{GM}{r^3}x = a_x$$

$$\frac{d^2y}{dt^2} = -\frac{GM}{r^3}y \qquad \Rightarrow \qquad \frac{dv_y}{dt} = -\frac{GM}{r^3}y = a_y$$

$$\frac{dx}{dt} = v_x$$

$$\frac{dy}{dt} = v_y$$

→一階連立微分方程式

一階微分方程式を解く数値解法を用いて解ける

一階微分方程式の数値解法

- 微分方程式の初期値問題

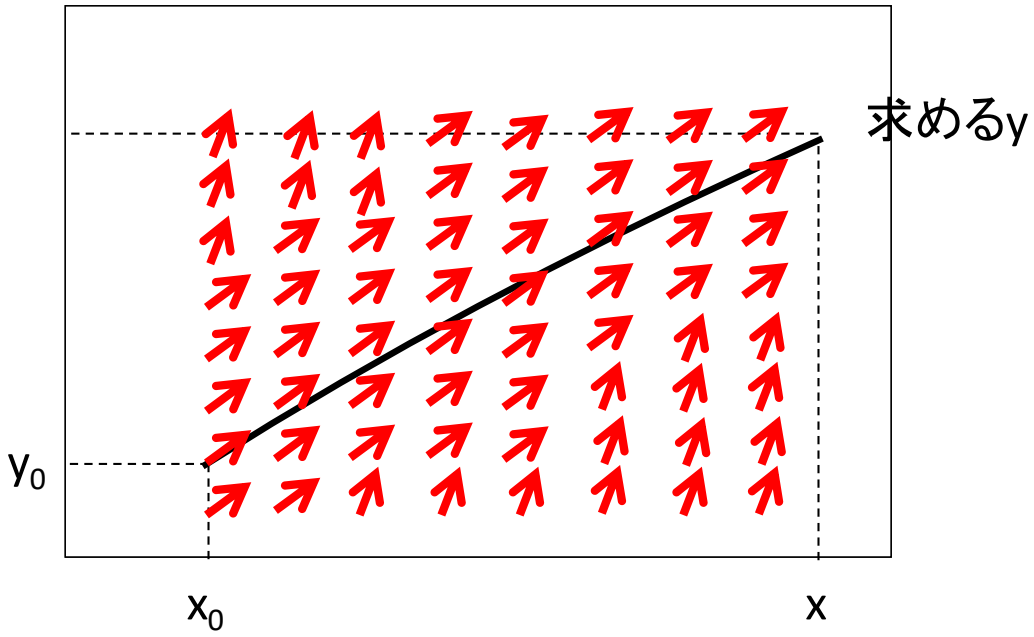
$$y'(x) \equiv \frac{dy}{dx} = f(x, y(x))$$

初期条件

$y=y_0$ @ $x=x_0$

指定されたx のときの　y　を求める

f(x,y(x))→各位置での曲線の傾き



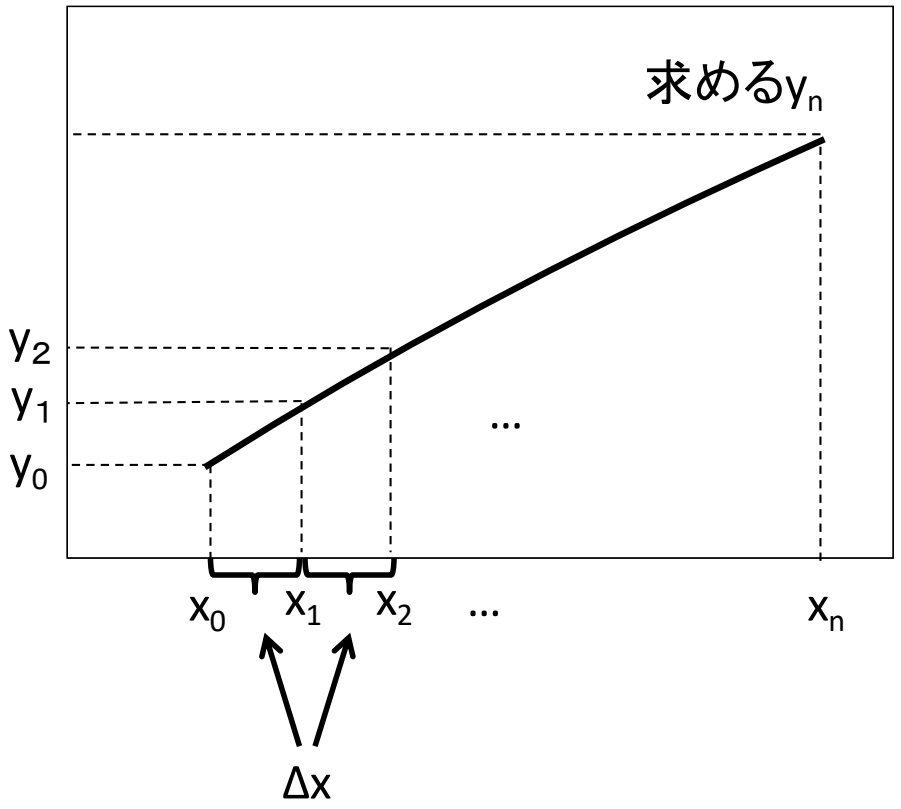
離散化

★離散化

連続的なものを離散的に表現

→xは一定の刻み幅（Δx）で離散的に変化すると仮定

→x_0,y_0からいきなりyを求めるのではなく、刻み幅毎に順次yを求めてゆく



〇計算手順

x_0,y_0→x_1(=x_0+Δx)でのy(=y_1)

x_1,y_1→x_2(=x_1+Δx)でのy(=y_2)

…

x_{n-1},y_{n-1}→x_n(=x_{n-1}+Δx)でのy(=y_n)

◎必要な計算手法

x_i,y_i からy_{i+1}を求める方法

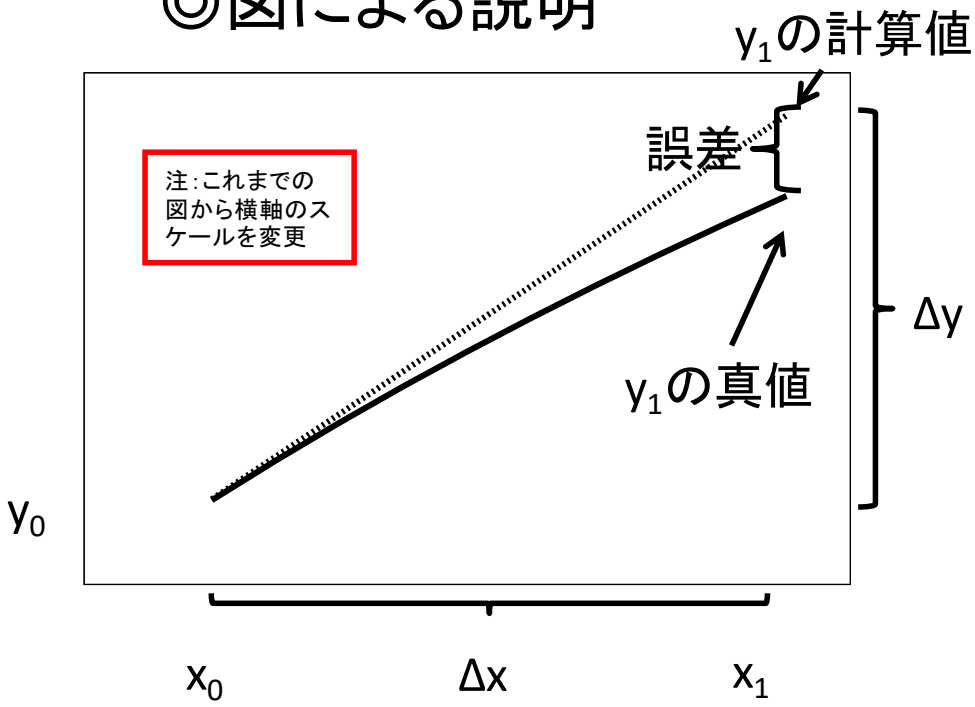
オイラー法(線形近似法)

- 微分方程式の初期値問題を数値的に解く手法の一種
  （最も簡単、計算精度に問題が出る場合あり）
- 考え方

$x=x_0$～x_1では傾き(f)は一定

この傾きは、$f(x_0,y_0)$で近似

◎図による説明



◎式の導出

$x=x_0$ で $y=y_0$ と仮定

x_0からx_1までの変化量は、x_0での傾きから計算

$$y(x_1) = y(x_0) + \Delta y \approx y(y_0) + f(x_0, y_0)\Delta x$$

★真値と計算値の差→誤差

Δxが小さいと誤差が減少すると期待

→小さなΔxの使用が必要

オイラー法(線形近似法)2

◎式による説明

$x=x_0$ で $y=y_0$ と仮定
$y(x_1)=y(x_0+\Delta x)$としてテーラー展開

$$y(x_1) = y(x_0 + \Delta x)$$

$$= \quad y(x_0) + y'(x_0)\Delta x + y''(x_0)\frac{\Delta x^2}{2!} + y'''(x_0)\frac{\Delta x^3}{3!}\cdots$$

$$= \quad y(x_0) + f(x_0, y_0)\Delta x + f'(x_0, y_0)\frac{\Delta x^2}{2!} + f''(x_0, y_0)\frac{\Delta x^3}{3!}\cdots$$

$$\approx \quad y(x_0) + f(x_0, y_0)\Delta x$$

（Δx^2以降の項を打ち切る）

→前頁で図の説明から得られた式と同じ式

微分方程式を解く簡単な例

◎dy/dx=2x の数値解

初期値 $x_0=1$, $y_0=1$

→ x=2 での y の値

〇解析解

$$\frac{dy}{dx}=2x \Rightarrow dy=2xdx \Rightarrow \int dy=\int 2x\,dx$$

$$\Rightarrow \left[y\right]_{y0}^{y1}=\left[x^2\right]_{x0}^{x1} \Rightarrow y_1-1=x_1^2-1 \Rightarrow y_1=x_1^2$$

$y=x^2$ より y=4.0 @ x=2

〇数値計算
オイラー法の計算式

$$y(x_{i+1})=y(x_i)+\Delta y_i \approx y(x_i)+f(x_i,y_i)\Delta x$$

$$or$$

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

計算結果：Δx=0.1 → n(=分割数)=10

n	x_n	y_n	$g(x)=2x$	y_n + 傾き × 0.10
0	1.00	1.00	2.00	$1.00+2.00\times 0.10=1.20$
1	1.10	1.20	2.20	$1.20+2.20\times 0.10=1.42$
2	1.20	1.42	2.40	$1.42+2.40\times 0.10=1.66$
3	1.30	1.66	2.60	$1.66+2.60\times 0.10=1.92$
4	1.40	1.92	2.80	$1.92+2.80\times 0.10=2.20$
5	1.50	2.20	3.00	$2.20+3.00\times 0.10=2.50$
6	1.60	2.50	3.20	$2.50+3.20\times 0.10=2.82$
7	1.70	2.82	3.40	$2.82+3.40\times 0.10=3.16$
8	1.80	3.16	3.60	$3.16+3.60\times 0.10=3.52$
9	1.90	3.52	3.80	$3.52+3.80\times 0.10=3.90$
10	2.00	3.90		

上記の問題をオイラー法で解くための反復計算式

$$\frac{dy}{dx}=2x\equiv f(x) \quad \text{より}$$

$$y(x_{i+1})=y(x_i)+2x_i\Delta x$$

$$or \qquad y_{i+1}=y_i+2x_i\Delta x$$

プログラム：微分方程式の例

```
#include <stdio.h>
#include <math.h>

void initial(double *x, double *y, double *dx, double
*xmax)
{
  *x=1.0;
  *y=1.0;
  *dx=0.1;
  *xmax=2.0;
  return;
}

void euler(double *x, double *y, double dx)
{
  double slope, change;

  slope = 2* *x;
  change = slope*dx;
  *y += change;
  *x += dx;
  return;
}

main()
{
  int    i, n, imax;
  double x, y, dx, xmax;

  initial(&x, &y, &dx, &xmax);
  i=0;
  imax=xmax/dx;
  printf("%4d %6.3f %6.3f¥n", i, x, y);
  while ( i < imax ){
    euler(&x, &y, dx);
    i++;
    printf("%4d %6.3f %6.3f¥n", i, x, y);
  }
}
```

注：このプログラムでは配列は使っていない

今回の問題をオイラー法で解くための
反復計算式

$$y_{i+1} = y_i + 2x_i\Delta x$$

計算結果：微分方程式の例

Δ x=0.1の場合

step	x	y
0	1.000	1.000
1	1.100	1.200
2	1.200	1.420
3	1.300	1.660
4	1.400	1.920
5	1.500	2.200
6	1.600	2.500
7	1.700	2.820
8	1.800	3.160
9	1.900	3.520
10	2.000	3.900

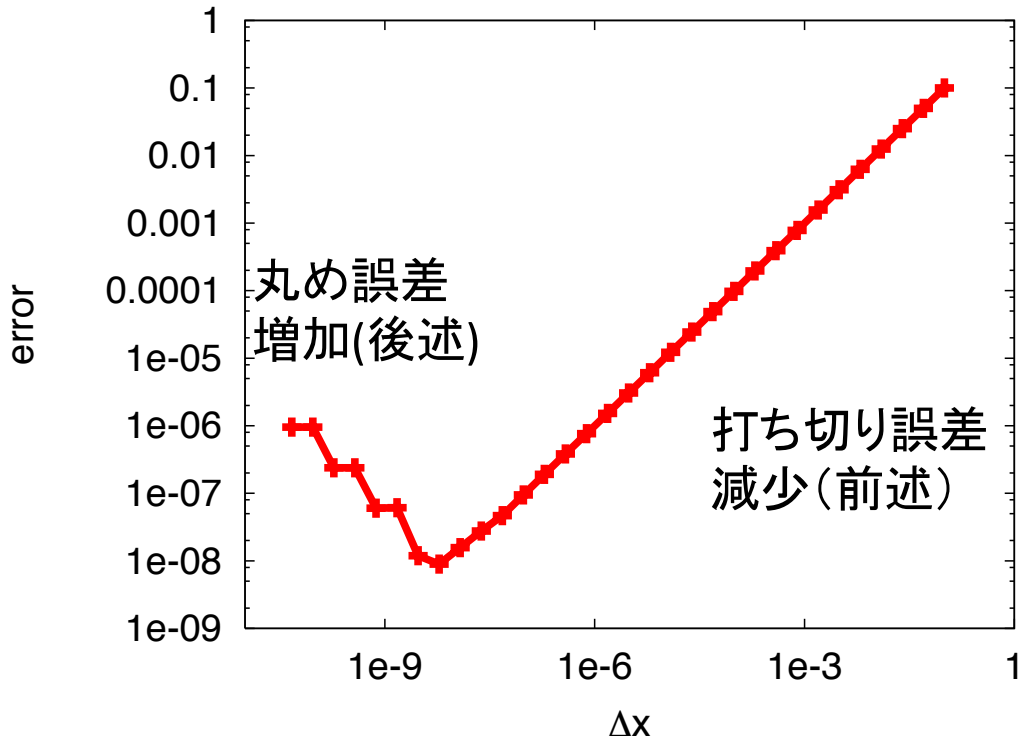
誤差0.1

Δ x=0.05の場合

step	x	y
0	1.000	1.000
1	1.050	1.100
2	1.100	1.205
3	1.150	1.315
4	1.200	1.430
5	1.250	1.550
6	1.300	1.675
7	1.350	1.805
8	1.400	1.940
9	1.450	2.080
10	1.500	2.225
11	1.550	2.375
12	1.600	2.530
13	1.650	2.690
14	1.700	2.855
15	1.750	3.025
16	1.800	3.200
17	1.850	3.380
18	1.900	3.565
19	1.950	3.755
20	2.000	3.950

誤差0.05

刻み幅による誤差の変化



- Δxの減少により、最初誤差が減少するが、その後増加

計算の精度

精度を考える時の実際的な指針

〇打切誤差

解法のアルゴリズムに依存

オイラー法では傾き一定と仮定したことによる誤差

実用上は結果が変化しなくなるまで Δt を小さくすることで対処

Δt を小さくし過ぎると丸め誤差の影響

但し Δt を小さくすると計算時間が増大 （計算時間の問題）

→ 別のアルゴリズムが必要

〇丸め誤差

実数の計算機内部での表現方法による問題

有効桁数：単精度 7 桁、倍精度 15桁

計算を繰り返すと丸め誤差が累積

同程度の数の引き算で桁落ちが発生

◎有効桁4桁の場合

- 丸め誤差の例

  1.000

  +0.0001

  =1.0001 → 1.000

- 桁落ちの例

5*6.403-32の場合

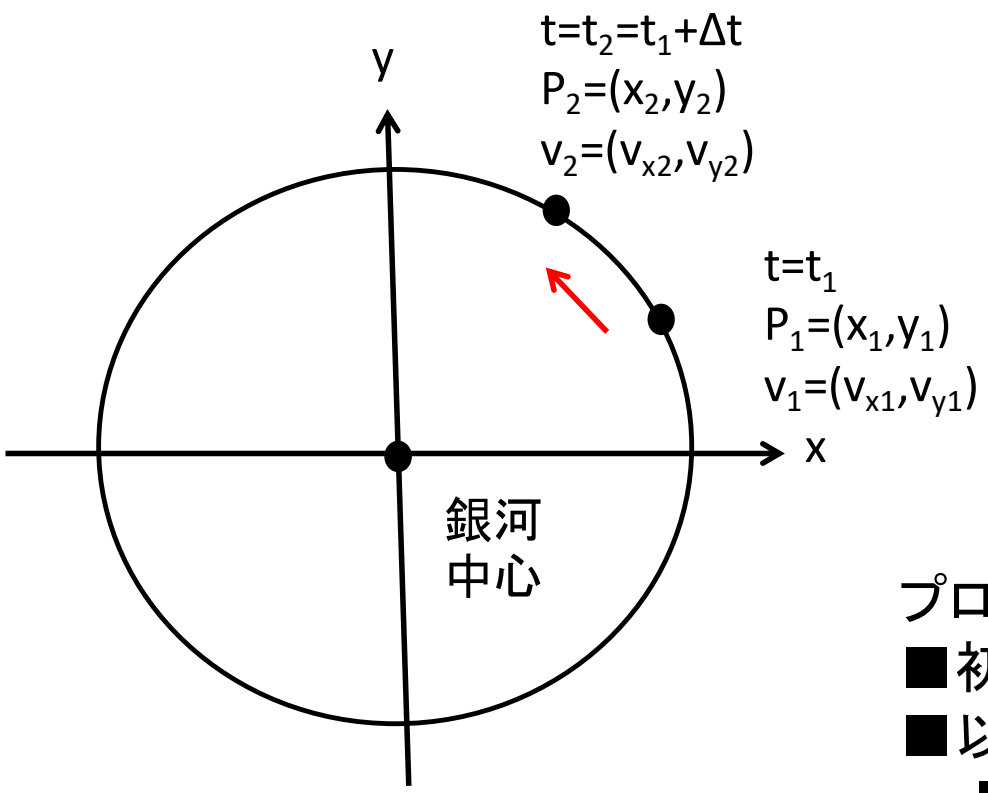
5*6.403=32.015→ 32.02

 32.02

-32.00

= 0.02 (有効桁1桁)

離散化:太陽の運動の場合



数値計算でよく用いられる方法

ここでは時間を離散化

時間は本来連続的に進むが、時間刻みΔt毎に離散的に進むと仮定

→太陽も連続的ではなく、離散的に移動

プログラムの流れ

■初期値設定

■以下を繰り返す

■Δt後の太陽の位置と速度を求める

★プログラム作成に必要なこと

時刻$t=t_1$の太陽の位置と速度から

時刻$t=t_2$の太陽の位置と速度を求める方法

オイラー法による太陽の運動の計算式

$$a_x(t_1) = -\frac{GM}{r(t_1)^3} x(t_1), \qquad \left(r(t_1) = \sqrt{x(t_1)^2 + y(t_1)^2} \right)$$

$$a_y(t_1) = -\frac{GM}{r(t_1)^3} y(t_1),$$

$$v_x(t_2) = v_x(t_1) + a_x(t_1) \times \Delta t,$$

$$v_y(t_2) = v_y(t_1) + a_y(t_1) \times \Delta t,$$

$$x(t_2) = x(t_1) + v_x(t_1) \times \Delta t,$$

$$y(t_2) = y(t_1) + v_y(t_1) \times \Delta t,$$

但し、太陽の公転の場合、オイラー法では動かない
→オイラー•クロマー法が必要

オイラー•クロマー法による太陽の運動の計算式

$$a_x(t_1) = -\frac{GM}{r(t_1)^3} x(t_1), \qquad \left(r(t_1) = \sqrt{x(t_1)^2 + y(t_1)^2} \right)$$

$$a_y(t_1) = -\frac{GM}{r(t_1)^3} y(t_1),$$

$$v_x(t_2) = v_x(t_1) + a_x(t_1) \times \Delta t,$$

$$v_y(t_2) = v_y(t_1) + a_y(t_1) \times \Delta t,$$

$$x(t_2) = x(t_1) + v_x(t_2) \times \Delta t,$$

$$y(t_2) = y(t_1) + v_y(t_2) \times \Delta t,$$

規格化#1

- M, m, r（G）は非常に大きな（小さな）数
  そのまま使用
  →結果が見にくくて、分かりにくい

- 時間と長さを小さな値に変換（規格化）
  時間：1億年→1
  長さ：1万光年→1
  →太陽の公転半径　3
  　　公転周期　2.26

☆問題点：GMは？

規格化#2

- 変換方法→円運動の性質を利用
  - 加速度と回転半径の関係

$$a = \frac{v^2}{r} \quad \rightarrow \quad m\frac{v^2}{r} = \frac{GMm}{r^2} \quad \rightarrow \quad v = \sqrt{\frac{GM}{r}}$$

  - 周期と回転半径の関係

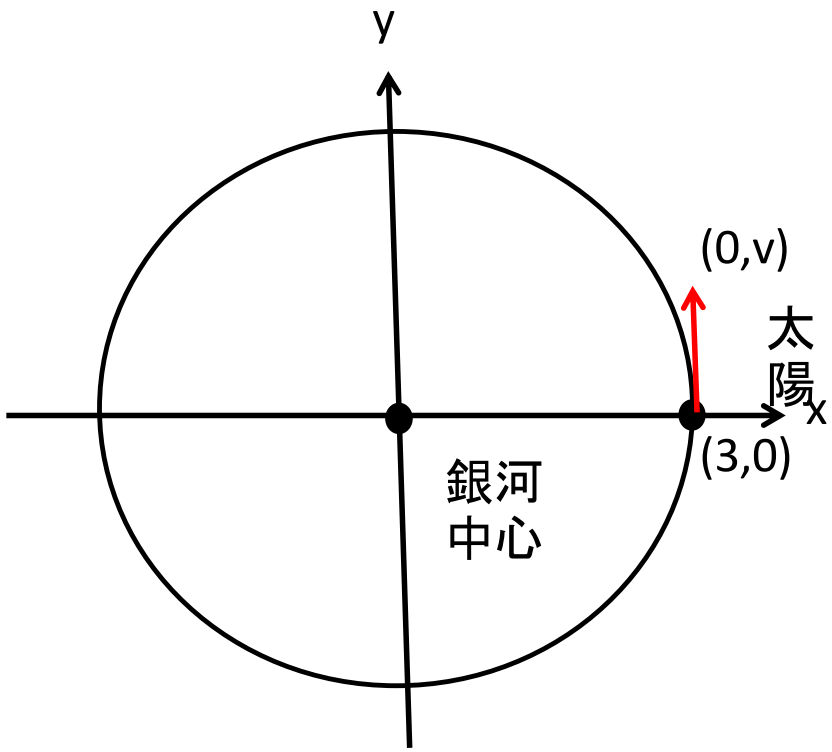
$$T = 2\pi\frac{r}{v}$$

  - 上記を組み合わせる

$$T^2 = \frac{4\pi^2}{GM}r^3 \quad \rightarrow \quad GM = \frac{4\pi^2 r^3}{T^2}$$

太陽の場合 r=3, T=2.26 より　GM=$4\pi^2 \times 3^2/2.26^2$

初期値設定



- 初期位置

どこでもよいが、分かりやすい位置が便利
例えばx軸上に配置
→ (3,0) : 3は太陽の公転半径

- 初期速度

初期位置が(3,0) の場合
→(0,v) : v は太陽の回転速度

$$v = \sqrt{\frac{GM}{r}}$$ より計算

プログラム概要

- 初期値設定

  太陽の初期位置と速度を設定

■ 以下を繰り返す

  - Δt後の太陽の位置と速度を求める

$$a_x = -\frac{GM}{r^3}x, \qquad a_y = -\frac{GM}{r^3}y,$$

$$v_x = v_x + a_x * \Delta t, \qquad v_y = v_y + a_y * \Delta t,$$

$$x = x + v_x * \Delta t, \qquad y = y + v_y * \Delta t,$$

問題2　銀河のシミュレーション

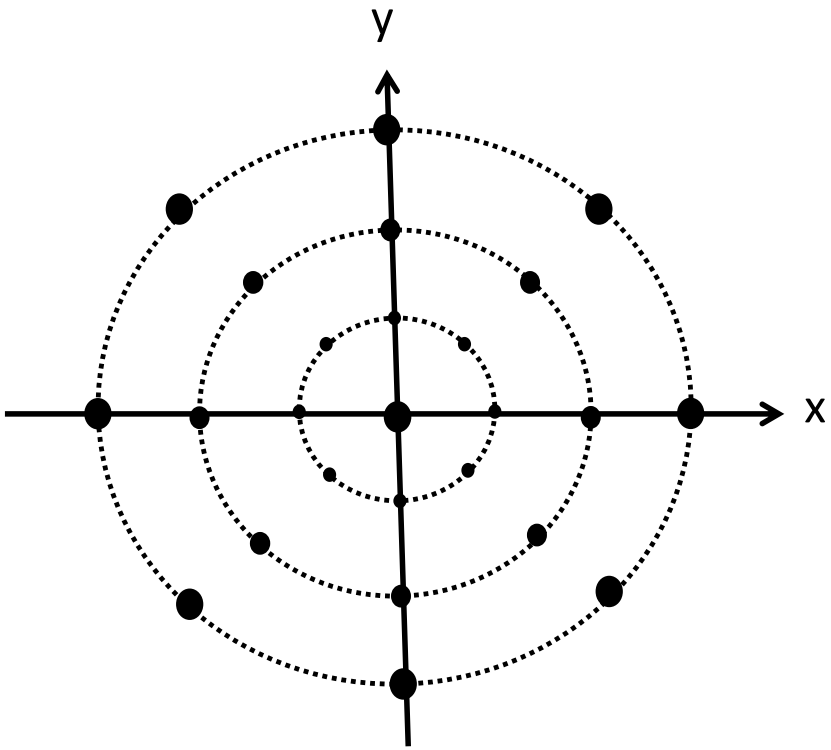
追加事項

- 恒星を増やす
  - 太陽と同じ質量の星
  - リング状に配置
- 恒星の初速度

$$v = \sqrt{\frac{GM}{r}}$$ より計算

- 恒星間の引力
  - 恒星の質量を銀河中心の質量の$1/10^{11}$と仮定

  →Gmを求める



プログラムの流れ

- 初期値設定

  全ての恒星の初期位置と速度を設定

■以下を繰り返す

- for(i=0 ; i < n ; i++)

  星iが全ての星から受ける力の和を計算

- for(i=0 ; i < n ; i++)

  星iの速度と位置の変化を計算

計算時間の測定

- time コマンドを使う方法
  - プログラム起動の時間を除く工夫が必要
    （計算回数を変えて2度計算し、その差から微分方程式を一度解く計算時間を見積もる）
- Cの関数を使う方法

→いずれの場合も、測定精度を上げる為に、計算時間に応じて計算回数を増やして測定を行い、得られた結果から微分方程式を一度解く計算時間を見積もる