今更聞けない数値計算アルゴリズム
常微分方程式 編

宮武 勇登

大阪大学

サイバーメディアセンター 准教授

2023年9月11日

今更聞けない数値計算アルゴリズム とは…

サイバーメディアセンターの新しいセミナーシリーズ

Q1 スパコンの利用に直結する講習会・セミナーは多いけれど、
一歩立ち戻って基礎的な内容のセミナーはない？

Q2 今使っているアルゴリズムの高速化も大事だけれど、
そもそももっと良いアルゴリズムはないのだろうか？

Q3 基礎的な内容ってどうすれば効率的に復習できる？

今更聞けない数値計算アルゴリズム とは...

- 第一弾として，**常微分方程式の数値解法**について

Q1 数値解法はどのような基準で選択すべき？

Q2 数値計算の入門書に書いてあることはだいたい同じ...
それだけで十分？

Q3 並列計算可能なの？

- 次回以降のためぜひフィードバックをおよせください！

自己紹介

- 微分方程式の**数値解析**

  応用数学の一分野としての **数値解析** とは
  問題を数値的に解析するための
  　アルゴリズム（数値解法）の開発
  およびその数学解析

- 数値線形代数

  連立一次方程式や行列関数など
  行列に関連するアルゴリズムの研究

- 不確実性定量化

- 地震分野などとの協働

略歴

2015	博士（情報理工学）東京大学
2015 - 2017	名古屋大学工学研究科 助教
2018 -	大阪大学サイバーメディアセンター 准教授 （情報科学研究科情報基礎数学専攻）

所属学会

日本応用数理学会，日本数学会，SIAM

今日の内容：常微分方程式の初期値問題

常微分方程式の初期値問題

$$\frac{\mathrm{d}}{\mathrm{d}t}u(t) = f\big(u(t)\big), \quad u(0) = u_0$$

例：Newtonの運動方程式

$$\frac{\mathrm{d}^2}{\mathrm{d}t^2}q(t) = f\big(q(t)\big)$$

1階のODEに書き換えられる

$$\begin{cases} \dfrac{\mathrm{d}}{\mathrm{d}t}q(t) = p(t) \\ \dfrac{\mathrm{d}}{\mathrm{d}t}p(t) = f(q(t)) \end{cases}$$

従属変数

$u(t)$

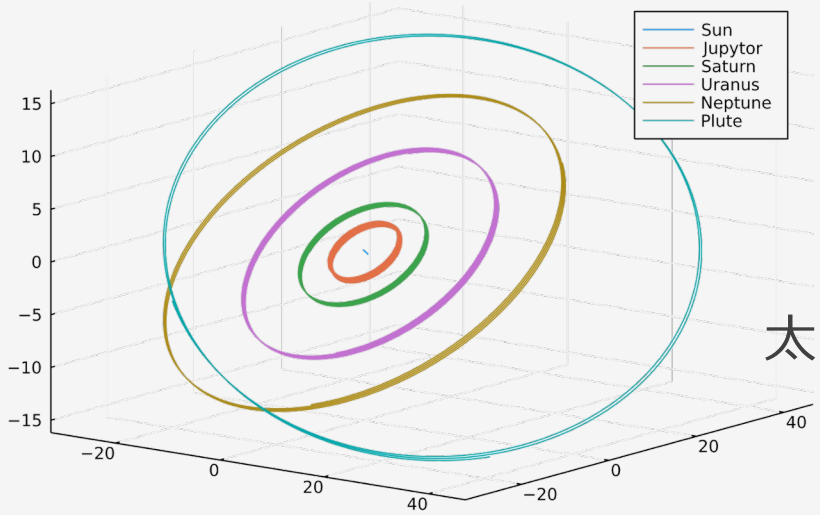
独立変数

注意

教科書・論文によっては

従属変数：
様々なアルファベット，太字，
まれにギリシャ文字

独立変数：
t の代わりに x が用いられることも



太陽系のシミュレーション

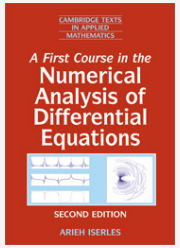
ODEの数値解法の研究の概観

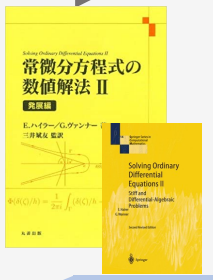
1700年　1900年　1960年　1990年　2022年

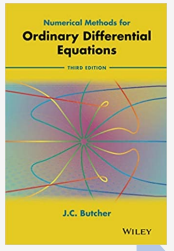
汎用解法

Newton　Euler　Runge Kutta　Dahlquist　Butcher　Wanner　Hairer…

学部レベルの数値計算の入門書　専門書でカバー

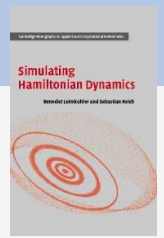


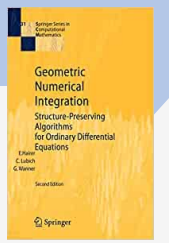


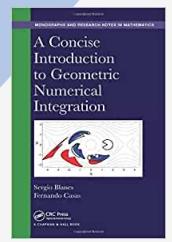


ODEの構造を活かした数値解法

Newton　Störmer　Verlet　Hairer, Sanz-Serna, Lubich, McLachlan…



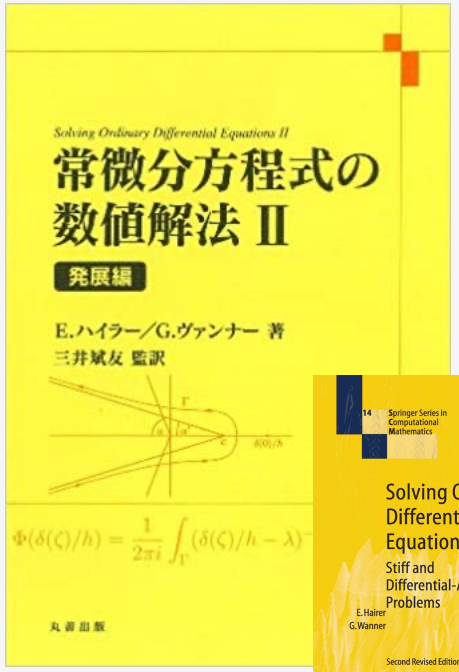


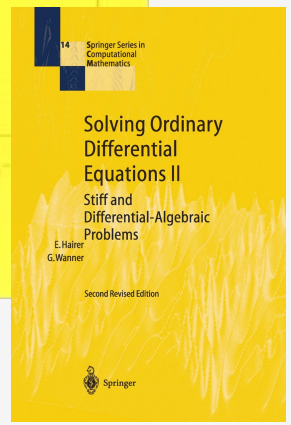


並列計算可能な数値解法

教科書のおすすめは？　注意点は？

- 汎用解法について



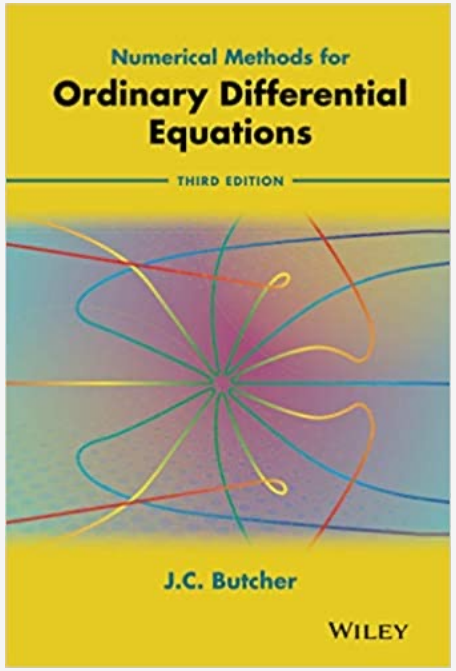


和書で、入門書よりは深く、かつ専門的過ぎない

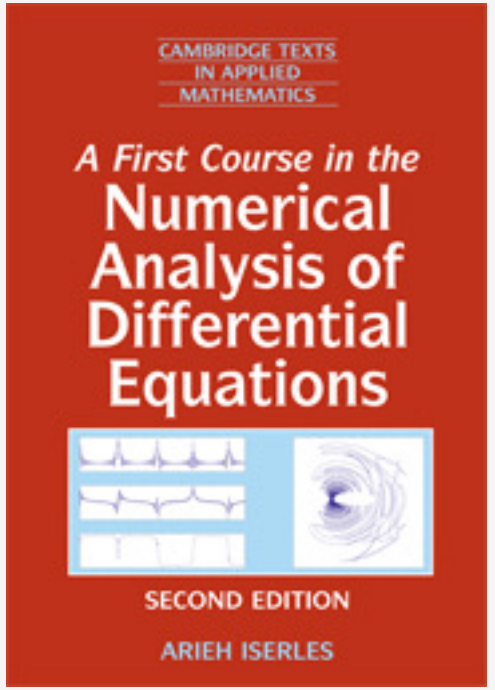
定番の教科書：1990年代半ばくらいまでの研究内容が網羅的に整理されている

教科書のおすすめは？　注意点は？

- 汎用解法について



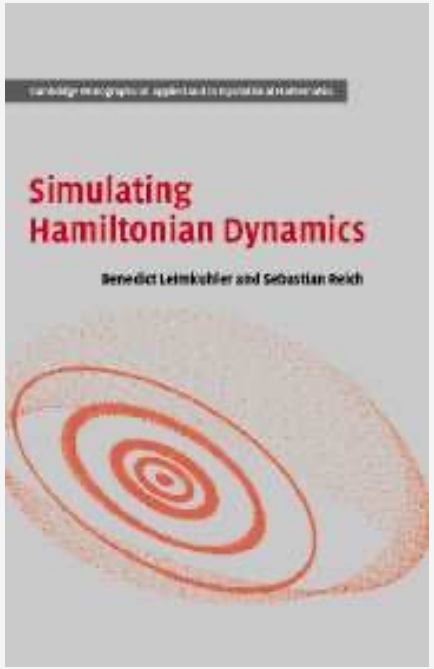
3rd edn. は，より最新の内容も含んでいる



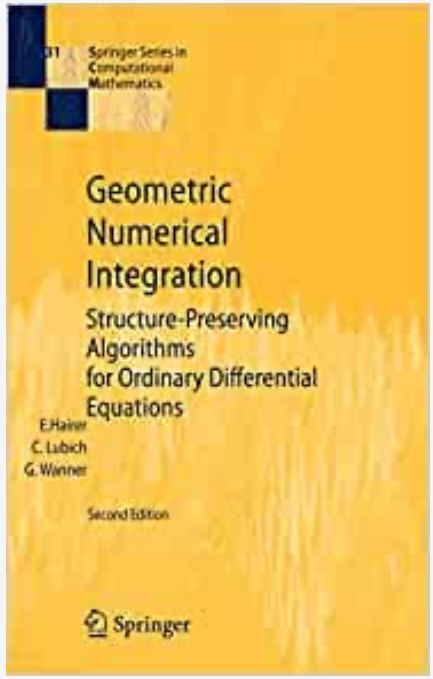
PDEも扱っている

教科書のおすすめは？　注意点は？

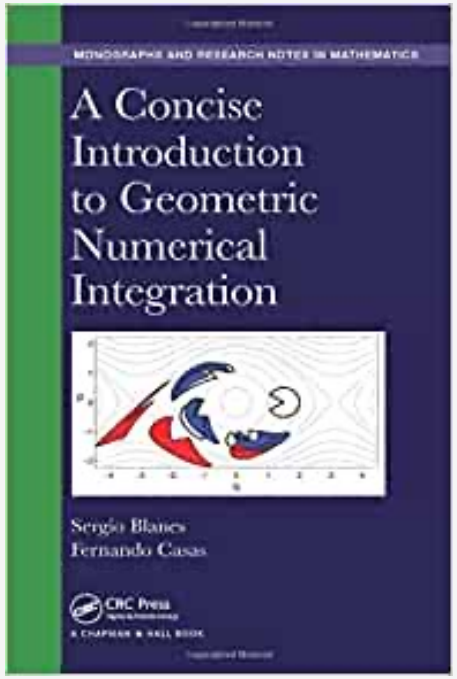
- 構造保存数値解法編



対象はHamilton系が中心
（やや古いのと対象が限定的）
数学的にはHLWほど高度ではなく読みやすい



定番！
数学的な難易度はやや高め



一番新しい
対象はやや限定的だが，読みやすい

常微分方程式の数値解法の分類（1/2）

$$\frac{\mathrm{d}}{\mathrm{d}t}u(t) = f\big(u(t)\big), \quad u(0) = u_0$$

基本的には**離散変数法**

離散時刻 $t_1, t_2, t_3, \ldots$ 上の近似解を求める

（刻み幅が等間隔の場合は $t_n = nh$）

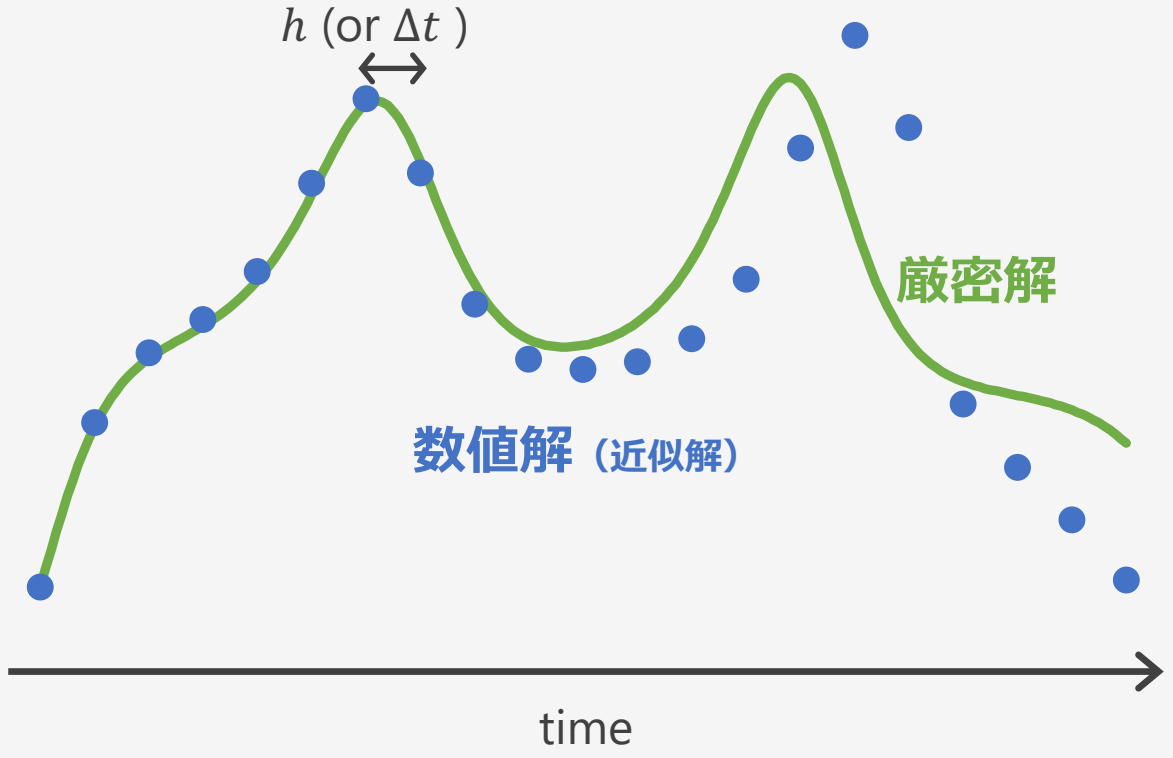
$$u_1 \approx u(t_1)$$

$$u_2 \approx u(t_2)$$

$$u_3 \approx u(t_3)$$

$$\vdots$$

刻み幅 h は一定でなくてもよい

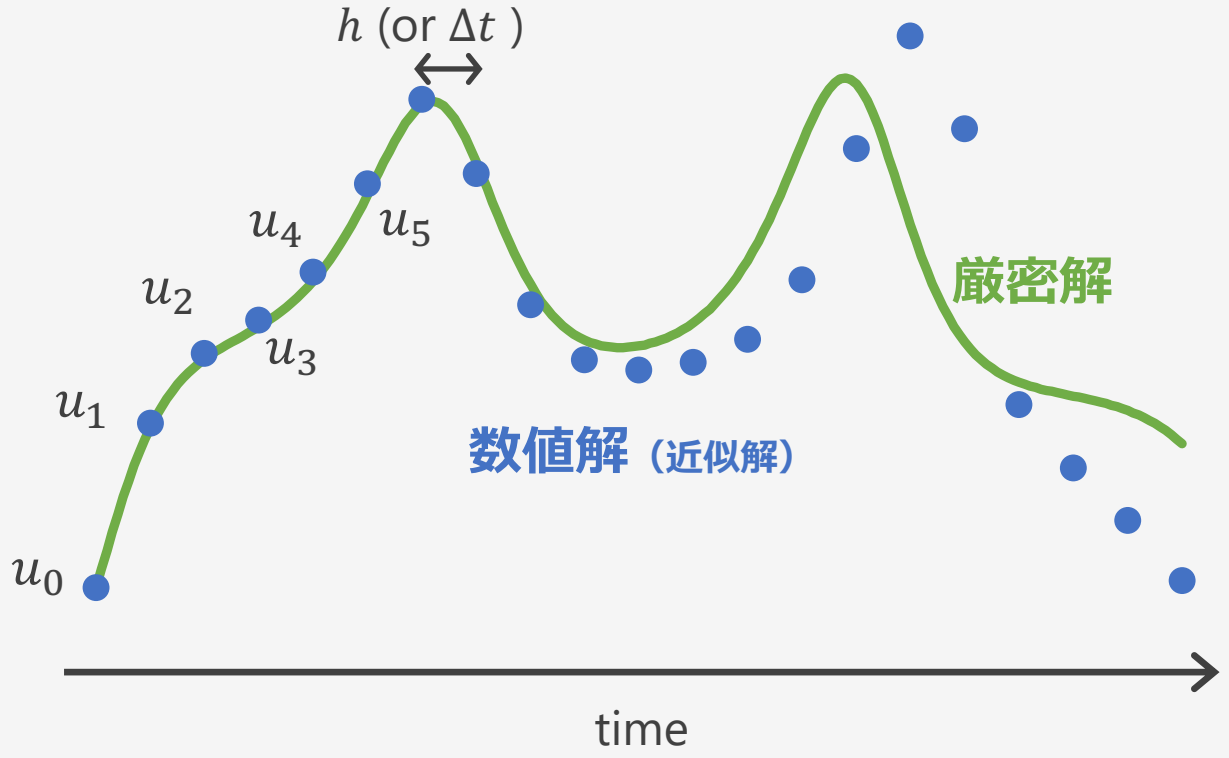


常微分方程式の数値解法の分類（2/2）

離散変数法の代表的解法

一段解法　u_n から u_{n+1} を計算

- Runge–Kutta法
- 分離型Runge–Kutta法
- splitting法，など



多値法　$u_{n-k}, \ldots, u_n$ から u_{n+1} を計算

- 線形多段解法
- 一般線形法，など

今日は主にRunge–Kutta法について話します

今日の内容

- 汎用解法について
  - Runge–Kutta法の導入
  - 数値解法の精度
  - 数値解法の安定性
  - その他（Runge–Kutta法以外）の代表的な数値解法クラス
- 構造保存数値解法
- ODEの並列計算（時間があれば）

Runge–Kutta法の導入

Euler法

$$\frac{\mathrm{d}}{\mathrm{d}t}u(t) = f\big(u(t)\big)$$

陽的Euler法

$$\lim_{h\to 0}\frac{u(t_n+h)-u(t_n)}{h} = f\big(u(t_n)\big)$$

$$\frac{u(t_n+h)-u(t_n)}{h} \approx f\big(u(t_n)\big)$$

$$\boxed{\frac{u_{n+1}-u_n}{h} = f(u_n)}$$

$$\left(\ u_{n+1} = u_n + hf(u_n)\ \right)$$

陽解法！

陰的Euler法

$$\lim_{h\to 0}\frac{u(t_{n+1})-u(t_{n+1}-h)}{h} = f\big(u(t_{n+1})\big)$$

$$\frac{u(t_{n+1})-u(t_n)}{h} \approx f\big(u(t_{n+1})\big)$$

$$\boxed{\frac{u_{n+1}-u_n}{h} = f(u_{n+1})}$$

$$\left(\ u_{n+1} = u_n + hf(u_{n+1})\ \right)$$

方程式を解く必要 ➡ 陰解法

積分の近似としてのEuler法の解釈

「微分の近似」だけではなく「**積分の近似**」としても理解できる

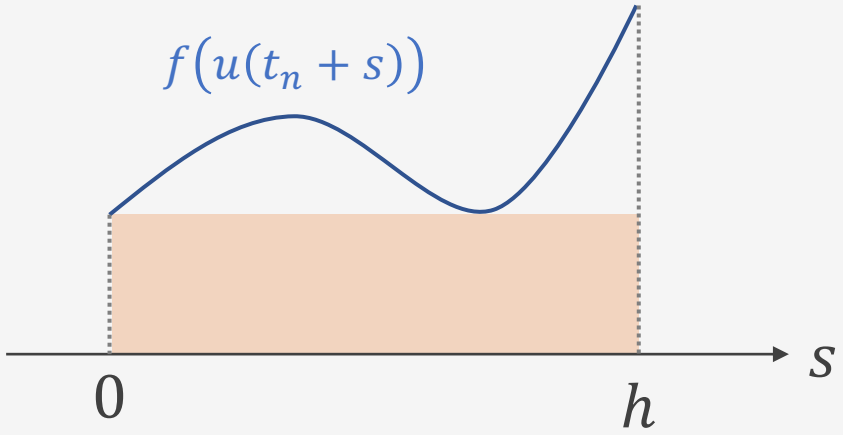
$\frac{\mathrm{d}}{\mathrm{d}t}u(t) = f\big(u(t)\big)$ の時間発展の積分表現：$u(t_{n+1}) = u(t_n) + \int_0^h f\big(u(t_n+s)\big)\mathrm{d}s$

（ODEの両辺を t_n から t_{n+1} まで積分）

$$u(t_{n+1}) = u(t_n) + \int_0^h f\big(u(t_n+s)\big)\mathrm{d}s$$

$$\approx u(t_n) + hf(u(t_n))$$

➡ 陽的Euler法　$u_{n+1} = u_n + hf(u_n)$



他の近似も可能！

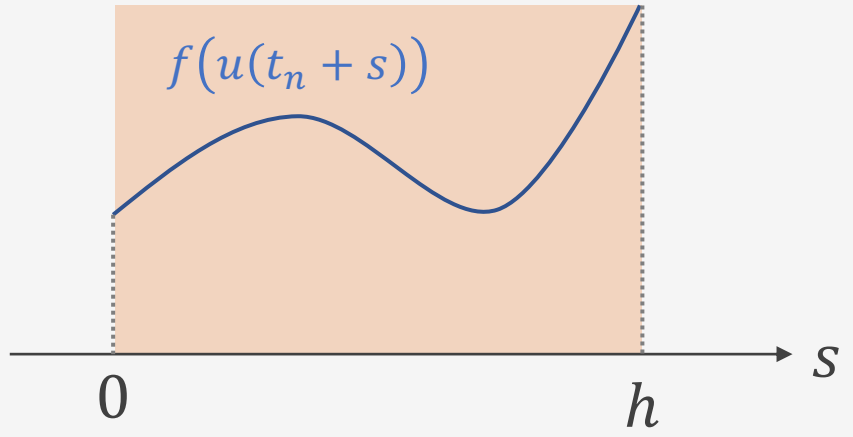
$$u(t_{n+1}) = u(t_n) + \int_0^h f\big(u(t_n + s)\big)\mathrm{d}s$$

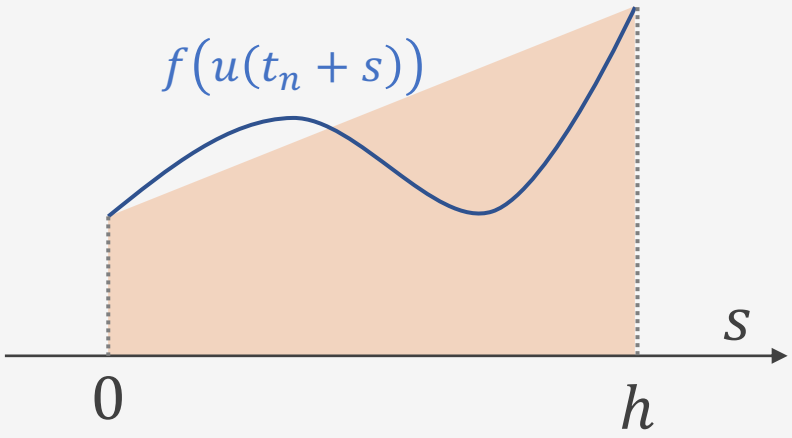
$$\approx u(t_n) + hf(u(t_{n+1}))$$

➡ 陰的Euler法　$u_{n+1} = u_n + hf(u_{n+1})$

$$\approx u(t_n) + h\frac{f\big(u(t_{n+1})\big) + f(u(t_n))}{2}$$

➡ 台形則　$u_{n+1} = u_n + h\dfrac{f(u_{n+1}) + f(u_n)}{2}$



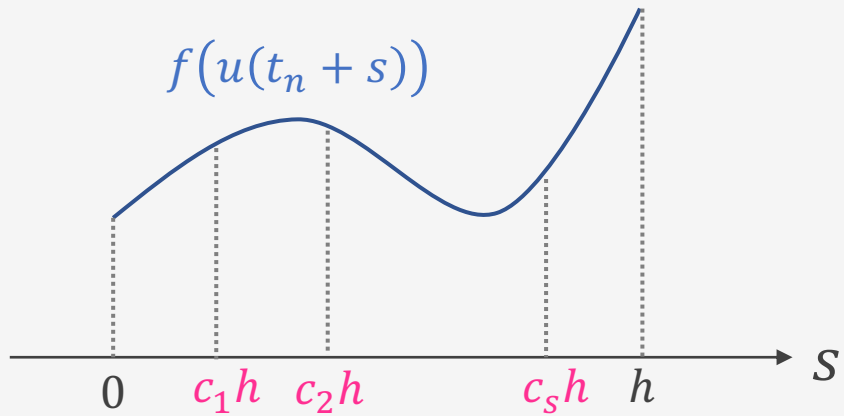


複数の内部点を使って近似してみよう

$$u(t_{n+1}) = u(t_n) + \int_0^h f\big(u(t_n+s)\big)\mathrm{d}s$$

$$\approx u(t_n) + h\sum_{i=1}^{s} b_i f\big(u(t_n + c_i h)\big) \qquad \left(\sum_{i=1}^{s} b_i = 1\right)$$

s 点の値を使って近似



$u(t_n + c_i h)$ も s 点の値を使って近似

$$u(t_n + c_i h) = u(t_n) + \int_0^{c_i h} f\big(u(t_n+s)\big)\mathrm{d}s$$

$$\approx u(t_n) + h\sum_{j=1}^{s} a_{ij} f\big(u(t_n + c_j h)\big)$$

$$\left(\sum_{j=1}^{s} a_{ij} = c_i\right)$$

注 意

実は，必ずしも内部の点でなくてもよい

$c_i = c_j$ $(i \neq j)$ という場合も考えられる

複数の内部点を使って近似してみよう

$$u(t_{n+1}) = u(t_n) + \int_0^h f\big(u(t_n+s)\big)\mathrm{d}s$$

$$\approx u(t_n) + h\sum_{i=1}^{s} b_i f\big(u(t_n + c_i h)\big) \qquad \left(\sum_{i=1}^{s} b_i = 1\right)$$

$u(t_n + c_i h)$ も s 点の値を使って近似

$$u(t_n + c_i h) = u(t_n) + \int_0^{c_i h} f\big(u(t_n+s)\big)\mathrm{d}s$$

$$\approx u(t_n) + h\sum_{j=1}^{s} a_{ij} f\big(u(t_n + c_j h)\big)$$

$$\left(\sum_{j=1}^{s} a_{ij} = c_i\right)$$

Runge–Kutta法

$$u_{n+1} = u_n + h\sum_{i=1}^{s} b_i f(U_i)$$

$$U_i = u_n + h\sum_{j=1}^{s} a_{ij} f(U_j) \quad (i = 1, \ldots, s)$$

Runge–Kutta法

$\dfrac{\mathrm{d}}{\mathrm{d}t}u(t) = f\big(u(t)\big)$　に対するRK法

Runge–Kutta法

$$u_{n+1} = u_n + h\sum_{i=1}^{s} b_i f(U_i) \quad \left(\sum_{i=1}^{s} b_i = 1\right)$$

$$U_i = u_n + h\sum_{j=1}^{s} a_{ij} f(U_j) \quad (i = 1, \dots, s)$$

U_i は $u(t_n + c_i h)$ の近似と解釈できる

ただし $c_i = \sum_{j=1}^{s} a_{ij}$

文献によっては，$k_i := f(U_i)$ として

$$u_{n+1} = u_n + h\sum_{i=1}^{s} b_i k_i$$

$$k_i = f\left(u_n + h\sum_{j=1}^{s} a_{ij} k_j\right)$$

補足：Runge–Kutta法（非自励系バージョン）

非自励系 $\frac{\mathrm{d}}{\mathrm{d}t}u(t) = f(u(t), t)$ の場合

c_i を陽に用いる！

Runge–Kutta法

$$u_{n+1} = u_n + h\sum_{i=1}^{s} b_i f(U_i, t_n + c_i h)$$

$$U_i = u_n + h\sum_{j=1}^{s} a_{ij} f(U_j, t_n + c_j h)$$

Runge–Kutta法

Runge–Kutta法の公式は $\{a_{ij}\}, \{b_i\}, \{c_i\}$ で特徴付けられる

Butcher配列 (Butcher tableau)

$$\begin{array}{c|c} \boldsymbol{c} & A \\ \hline & \boldsymbol{b}^{\top} \end{array} = \begin{array}{c|ccc} c_1 & a_{11} & \cdots & a_{1s} \\ \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & \cdots & a_{ss} \\ \hline & b_1 & \cdots & b_s \end{array}$$

$\{a_{ij}\}, \{b_i\}$ が本質！

陽的Euler法

$$\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array}$$

陰的Euler法

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}$$

Rungeの公式

$$\begin{array}{c|cc} 0 & & \\ 1/2 & 1/2 & \\ \hline & 0 & 1 \end{array}$$

Rungeの公式

$$\begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & 1/2 & 1/2 \end{array}$$

Aが下三角（対角は0）なら陽解法！

しばしば Heun 法と呼ばれるが，
発見したのはRunge (1895)

通常「Runge–Kutta法」と言ったらこれ！

4段4次の陽的RK法

0				
$1/2$	$1/2$			
$1/2$		$1/2$		
1			1	
	$1/6$	$2/6$	$2/6$	$1/6$

$$U_1 = u_n$$
$$U_2 = u_n + \frac{h}{2} f(U_1)$$
$$U_3 = u_n + \frac{h}{2} f(U_2)$$
$$U_4 = u_n + h f(U_3)$$
$$u_{n+1} = u_n + h\left(\frac{1}{6} f(U_1) + \frac{2}{6} f(U_2) + \frac{2}{6} f(U_3) + \frac{1}{6} f(U_4)\right)$$

発見したのは Kutta (1901)

Kutta (1901) は以下の4段4次公式も発見

実はこの公式の方が誤差は小さいが，↑の方が覚えやすく広く利用されている

0				
$1/3$	$1/3$			
$2/3$	$-1/3$	1		
1	1	-1	1	
	$1/8$	$3/8$	$3/8$	$1/8$

数値解法の精度

数値計算の誤差

誤差：各時刻における厳密解と数値解の差（の絶対値・ノルム）

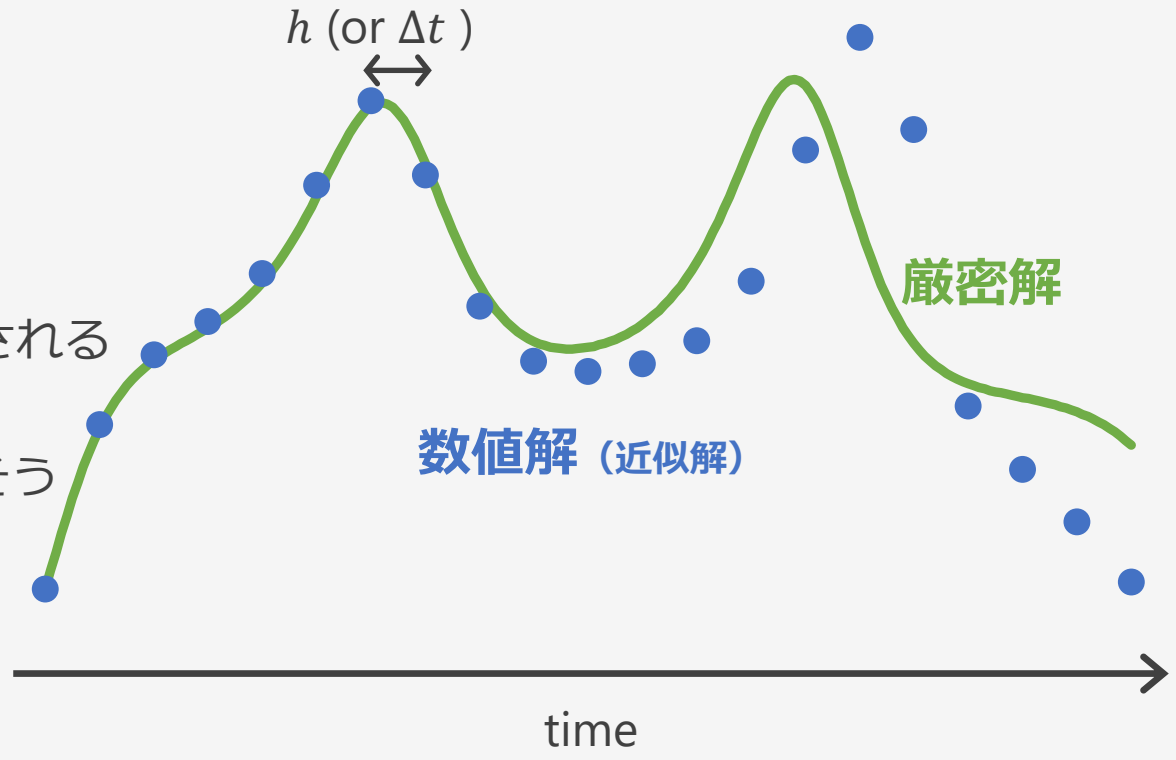
- 局所誤差：1ステップで混入する誤差
- 大域誤差：目標時刻における誤差（あるいは，目標時刻までの誤差の最大値）

誤差は時間発展に伴い蓄積されていく（ことが多い）

Q. どのように評価するか？

刻み幅 h を 0 に近づければ，誤差も小さくなることが期待される

誤差を h のべきで評価すれば，数値解法の性質が議論できそう



陽的Euler法の局所誤差

厳密解をTaylor展開

$$u(h) = u(0) + h\dot{u}(0) + \frac{h^2}{2}\ddot{u}(0) + \cdots$$

$$= u_0 + hf(u_0) + \frac{h^2}{2}f'(u_0)f(u_0) + \cdots$$

近似解をTaylor展開

$$u_1 = u_0 + hf(u_0)$$

$$\|u(h) - u_1\| = O(h^2)$$

h の1次の項まで合っている！

注意　$\frac{\mathrm{d}}{\mathrm{d}t}u(t) = f\big(u(t)\big), \quad u(0) = u_0$ に対して，

$$\dot{u}(0) = f\big(u(0)\big) = f(u_0) \qquad \ddot{u}(t) = \frac{\mathrm{d}}{\mathrm{d}t}\dot{u}(t) = \frac{\mathrm{d}}{\mathrm{d}t}f\big(u(t)\big) = f'(u(t))f(u(t))$$

精度

ODEの数値解法の精度

以下のように評価できる数値解法を「p 次（精度）の解法」と呼ぶ

$$\|u(h) - u_1\| = \mathrm{O}(h^{p+1})$$

大域誤差は…

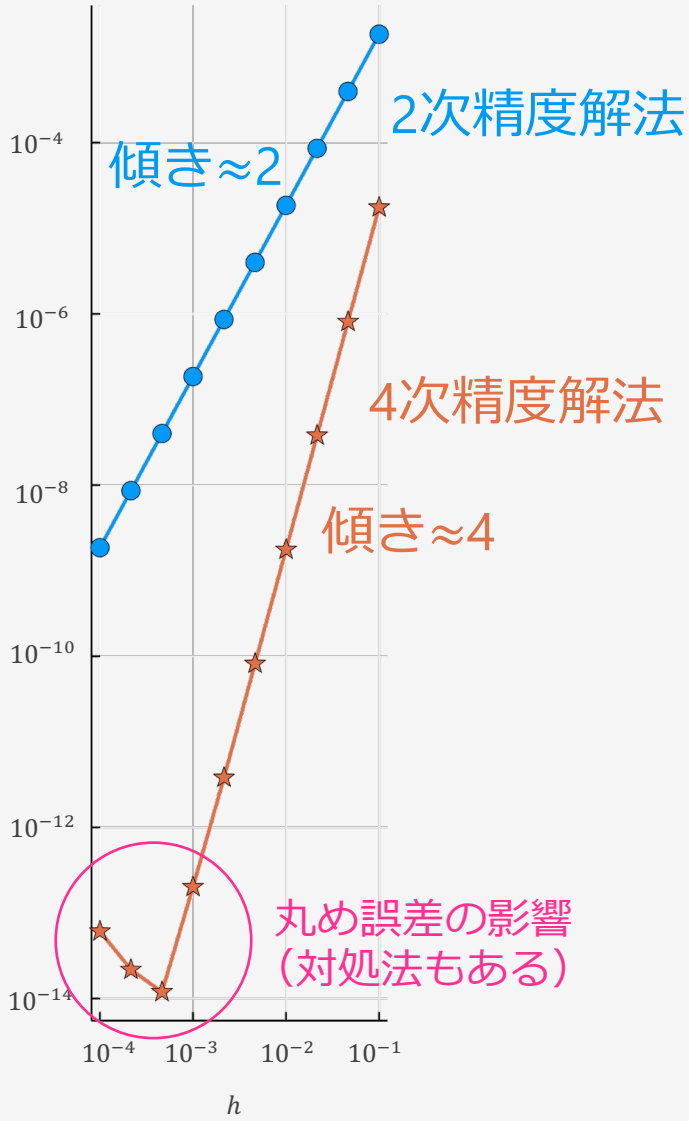
とてもラフに議論すれば，目標時刻を T としたとき， $t_N = T$ $(Nh = T)$における大域誤差は

$$Ch^{p+1} \times N = CNhh^p = CTh^p = \mathrm{O}(h^p)$$

非常にラフな議論だが，だいたい正しいことが多い

次数が違うと…

異なる刻み幅で計算したときの
大域誤差



- **次数と傾きがほぼ同じ**
- 高精度な数値解を得たい場合は，
  次数の高い解法のほうが効率的なことが多い
- 刻み幅を変えると誤差がいまと比べて
  相対的にどう変わるかが予想できる
- テスト問題を除いて、実際の誤差の大きさは
  あまりよく分からないことが多い

陽的RK法の精度

陽的Euler法

0	0
	1

1次

Rungeの公式

0		
$1/2$	$1/2$	
	0	1

2次

Rungeの公式

0		
1	1	
	$1/2$	$1/2$

2次

0				
$1/2$	$1/2$			
$1/2$		$1/2$		
1			1	
	$1/6$	$2/6$	$2/6$	$1/6$

4次

0				
$1/3$	$1/3$			
$2/3$	$-1/3$	1		
1	1	-1	1	
	$1/8$	$3/8$	$3/8$	$1/8$

4次

陽的RK法でp次精度を達成する最小段数

p	1	2	3	4	5	6	7	8	$\cdots$	10
最小段数	1	2	3	4	6	7	9	11	$\cdots$	17

- $p = 1$: 陽的Euler法
- $p = 2$: Runge (1895)
- $p = 3$: Heun (1900)
- $p = 4, 5$: Kutta (1901)

  ただし5次精度解法にはミスがあり

  Nyström (1925) で修正

  5段5次が可能かどうかはこの時代には不明

  ➡ Butcher (1963頃) により不可能であることが示される

- $p = 6$: Butcher (1964)
- $p = 7$: Butcher (1965) で8段では不可能であることが示される

  9段の公式は1968年頃

  （Butcher含め複数の研究者が独立に発見）

- $p = 8$: Curtis (1970)

  10段では不可能なことはButcher (1985) による

- $p = 10$: Hairer (1978)

  16段で可能かどうかは未解決

陰的RK法の場合

s 段 $2s$ 次の陰的RK法が存在する

陰的Euler法

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}$$

1次

中点則

$$\begin{array}{c|c} 1/2 & 1/2 \\ \hline & 1 \end{array}$$

2次

$$\begin{array}{c|cc} \frac{1}{2}-\frac{\sqrt{3}}{6} & \frac{1}{4} & \frac{1}{4}-\frac{\sqrt{3}}{6} \\ \frac{1}{2}+\frac{\sqrt{3}}{6} & \frac{1}{4}+\frac{\sqrt{3}}{6} & \frac{1}{4} \\ \hline & 1/2 & 1/2 \end{array}$$

4次

Gauss法

$\{b_i\}$がGauss求積に対応

$$\begin{array}{c|ccc} \frac{1}{2}-\frac{\sqrt{15}}{10} & \frac{5}{36} & \frac{2}{9}-\frac{\sqrt{15}}{15} & \frac{5}{36}-\frac{\sqrt{15}}{30} \\ \frac{1}{2} & \frac{5}{36}+\frac{\sqrt{15}}{24} & \frac{2}{9} & \frac{5}{36}-\frac{\sqrt{15}}{24} \\ \frac{1}{2}+\frac{\sqrt{15}}{10} & \frac{5}{36}+\frac{\sqrt{15}}{30} & \frac{2}{9}+\frac{\sqrt{15}}{15} & \frac{5}{36} \\ \hline & \frac{5}{18} & \frac{4}{9} & \frac{5}{18} \end{array}$$

6次

Runge–Kutta法の実装

陽解法

難しいことは特にない

「内部段の計算では，一度確保した配列を使い回す」といった工夫程度

刻み幅の制御を行うこともある（次のスライド）

陰解法

時間ステップごとに **連立非線形方程式** を解かないといけない

Newton法，簡易Newton法，不動点反復…

連立一次方程式に対しては，係数行列の性質を検討し，適切なアルゴリズムを用いる

刻み幅制御

「～次精度の数値解法で計算したい！」より
「～桁くらいの精度で計算したい！」という需要の方が大きい

➡ 逆算して，数値解法や刻み幅を選択

刻み幅制御

異なる精度の2解法を同時に走らせ，その差を誤差の近似と解釈

- 十分小さい ➡ 次の時間発展の刻み幅を大きくする
- 大きすぎる ➡ 刻み幅を小さくしてやり直し

（設定したtoleranceと比較して…）

注 意 局所的に混入する誤差の制御であって，
大域的な誤差の制御ではない

埋め込み型RK法

Fehlberg (1969)

0						
$\frac{1}{4}$	$\frac{1}{4}$					
$\frac{3}{8}$	$\frac{3}{32}$	$\frac{9}{32}$				
$\frac{12}{13}$	$\frac{1932}{2197}$	$-\frac{7200}{2197}$	$\frac{7296}{2197}$			
1	$\frac{439}{216}$	-8	$\frac{3680}{513}$	$-\frac{845}{4104}$		
$\frac{1}{2}$	$-\frac{8}{27}$	2	$-\frac{3544}{2565}$	$\frac{1859}{4104}$	$-\frac{11}{40}$	
	$\frac{16}{135}$	0	$\frac{6656}{12825}$	$\frac{28561}{56430}$	$-\frac{9}{50}$	$\frac{2}{55}$
	$\frac{25}{216}$	0	$\frac{1408}{2565}$	$\frac{2197}{4104}$	$-\frac{1}{4}$	0

A は同じで2つの b を考える！

← **5次**

← **4次**

- matlab の ode45 や SciPy の dopri5 は Dormand–Prince公式 (1980)
- 5次精度公式の結果を近似解として採用

数値解法の安定性

安定性

安定性の評価基準はたくさんあるが，最も基本的なものを紹介

テスト方程式（複素数値のスカラーODE）：

$$\frac{\mathrm{d}}{\mathrm{d}t}u(t) = \lambda u(t), \quad u(0) = u_0 \in \mathbb{C}$$

λ は複素数で，その実部は**負**とする

$u(t) = e^{\lambda t}u_0$ なので $|u(t)| = \left|e^{\lambda t}u_0\right| < |u_0|$ となるが，

この性質がRK法による数値解でも成り立つかを考えてみよう

Euler法の場合

陽 的 Euler 法

$u_1 = (1 + h\lambda)u_0$ より $|u_1| < |u_0|$ となるには

$$|1 + h\lambda| < 1$$

h が大きいと，この不等式は成り立たない

陰 的 Euler 法

$u_1 = u_0 + h\lambda u_1$ より $u_1 = \frac{1}{1-h\lambda}u_0$

$h > 0$ ならば必ず $|u_1| < |u_0|$

無条件に安定！

Runge–Kutta法の場合

RK法の時間発展の一表現

$$u_1 = R(z)u_0 \quad (z = h\lambda)$$

$$R(z) = \frac{\det(I - zA + z\boldsymbol{e}\boldsymbol{b}^{\mathsf{T}})}{\det(I - zA)} \qquad (\boldsymbol{e} = [1, \dots, 1]^{\mathsf{T}})$$

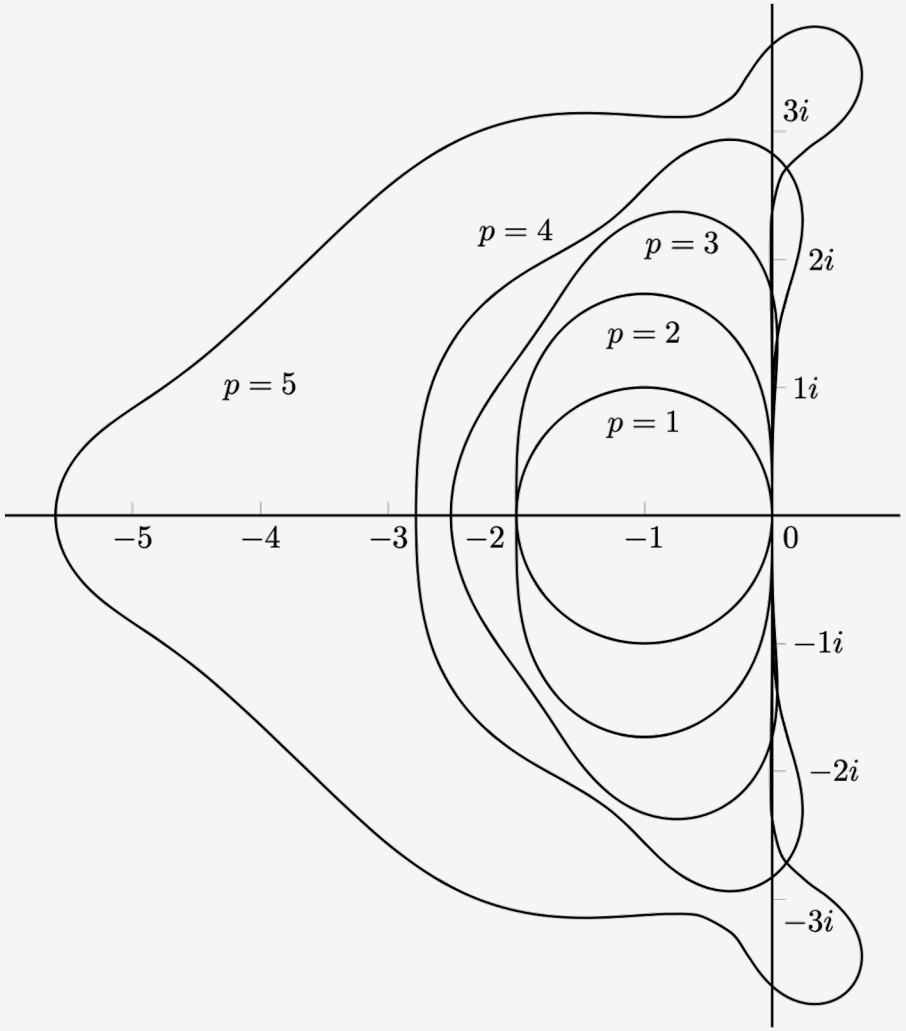
RK法の安定性因子

A安定性

- 安定性領域：$\mathcal{R} := \{z \in \mathbb{C} \mid |R(z)| < 1\}$
- $\mathcal{R} \supset \mathbb{C}^{-1} = \{z \in \mathbb{C} \mid \mathrm{Re}(z) < 0\}$ を満たす

RK法を「**A安定**な解法」という

陽的RK法の安定性領域



- p 段 p 次の陽的RK法の安定性領域は同一
  （例：4段4次の陽的RK法は複数存在するが，
  その安定性領域はどれも同じ）
- $p = 5$ (6段) の図は一例
- どれも $\mathbb{C}^{-1}$ からは程遠い…
  λ の実部が負の方に大きいと，
  h はものすごく小さくとらないといけない
- Gauss–Runge–Kutta法（陰解法）はA安定

硬い系 (stiffness)

硬い系：

刻み幅を極めて小さくとらないと，数値的に不安定になってしまう系

RK法の場合

陰解法を用いる

安定性領域ができる限り実軸の左の方を覆うように設計された陽解法を用いる

例題

例 1

$$\frac{\mathrm{d}}{\mathrm{d}t}u(t) = \begin{bmatrix} -2 & 1 \\ 2 & -3 \end{bmatrix} u(t) + \begin{bmatrix} \cos t \\ 3\cos t - \sin t \end{bmatrix}, \qquad u(0) = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

例 2

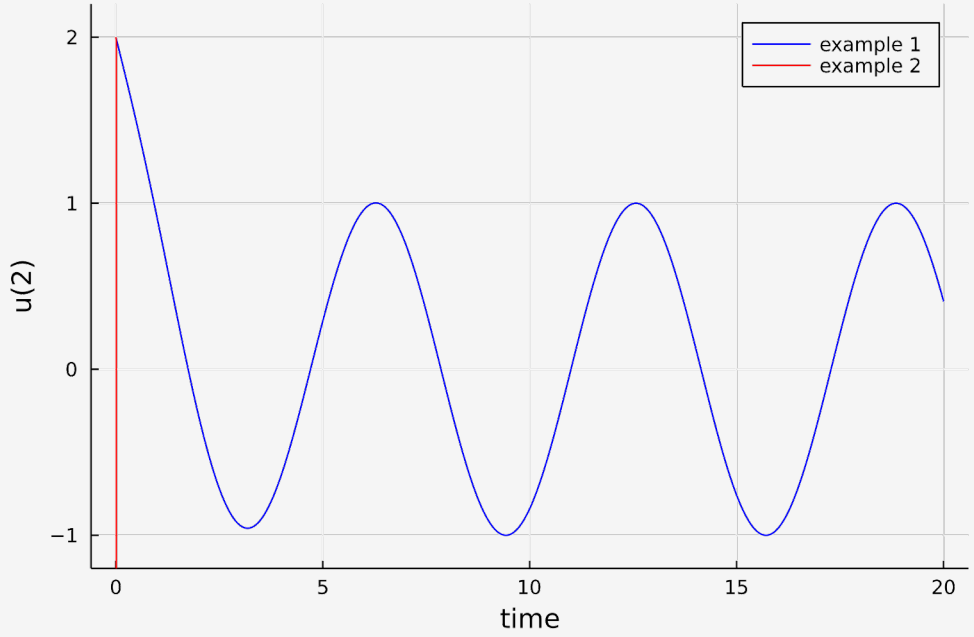
$$\frac{\mathrm{d}}{\mathrm{d}t}u(t) = \begin{bmatrix} -2 & 1 \\ 1998 & -1999 \end{bmatrix} u(t) + \begin{bmatrix} -\cos t \\ 1999\cos t - \sin t \end{bmatrix}, \qquad u(0) = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

実はどちらも解は　$u(t) = \exp(-t)\begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ \cos t \end{bmatrix}$

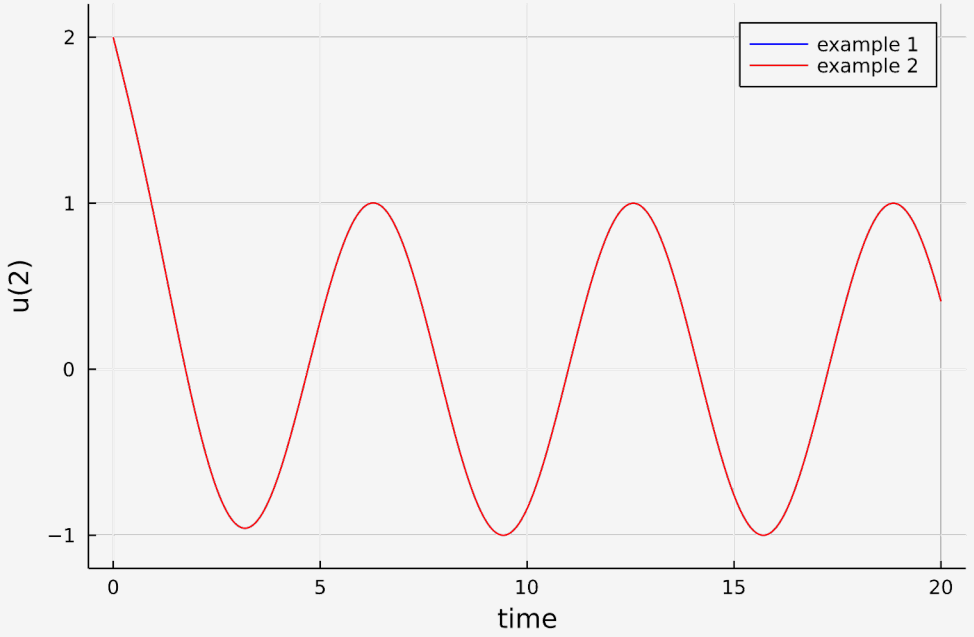
微分方程式もなんとなく似通っているし，RK法で計算したときの近似解も似ているはず…？

4段4次陽的RK法 と 中点則

4段4次の陽的RK法



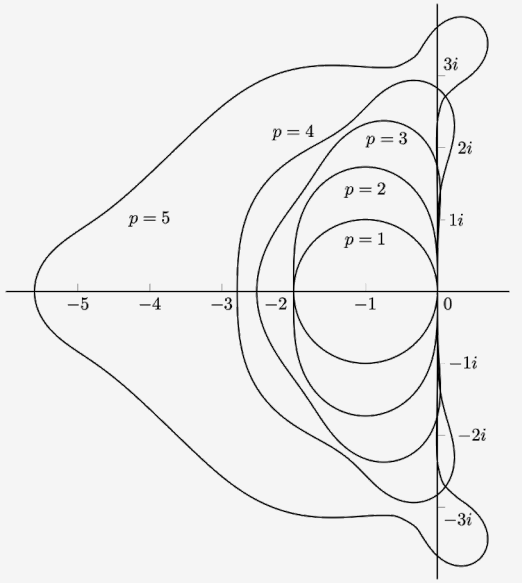
中点則



例2 は 2スッテプ目 で発散...

刻み幅はどちらも $h = 0.01$

2つの例の違いは何か？



例 1

$$\frac{\mathrm{d}}{\mathrm{d}t}u(t) = \underbrace{\begin{bmatrix} -2 & 1 \\ 2 & -3 \end{bmatrix}}u(t) + \begin{bmatrix} \cos t \\ 3\cos t - \sin t \end{bmatrix}, \qquad u(0) = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

固有値：$-1, -4$

だいたい $-3 < -4h$ $(h < 3/4)$ で安定的に計算できる

例 2

$$\frac{\mathrm{d}}{\mathrm{d}t}u(t) = \underbrace{\begin{bmatrix} -2 & 1 \\ 1998 & -1999 \end{bmatrix}}u(t) + \begin{bmatrix} -\cos t \\ 1999\cos t - \sin t \end{bmatrix}, \qquad u(0) = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

固有値：-1, -2000

$h < 3/2000$ くらい小さく取らないといけない

中点則なら特に制限なし（ただし精度の問題はある）

硬い系とは何か？

線形系：$\dfrac{\mathrm{d}}{\mathrm{d}t}u(t) = Au(t) + \phi(t)$ に対して

- 行列Aについて絶対値が大きな固有値を持つ問題は「**硬い系**」になる
- 硬度比が大きな問題も「硬い系」になりうる
- 硬度比が大きくても硬い系ではないかもしれないが
  初期値の変動に対しては不安定

$$\text{硬度比} = \frac{|A\text{の絶対値最大固有値}|}{|A\text{の絶対値最小固有値}|}$$

例2 は「硬い系」とまでは言えず，「**硬い系に近い**」という程度の問題

（注意）非線形問題の場合は，線形化して現れるJacobi行列の固有値に着目する

硬い系に対する数値解法

安定性領域が（特に左方向に）広い解法を使う！

- A安定な解法
- 硬い系専用の解法（matlab の ode23s など）
- 陽解法ならば，安定性領域が実軸左方向に広がっている解法

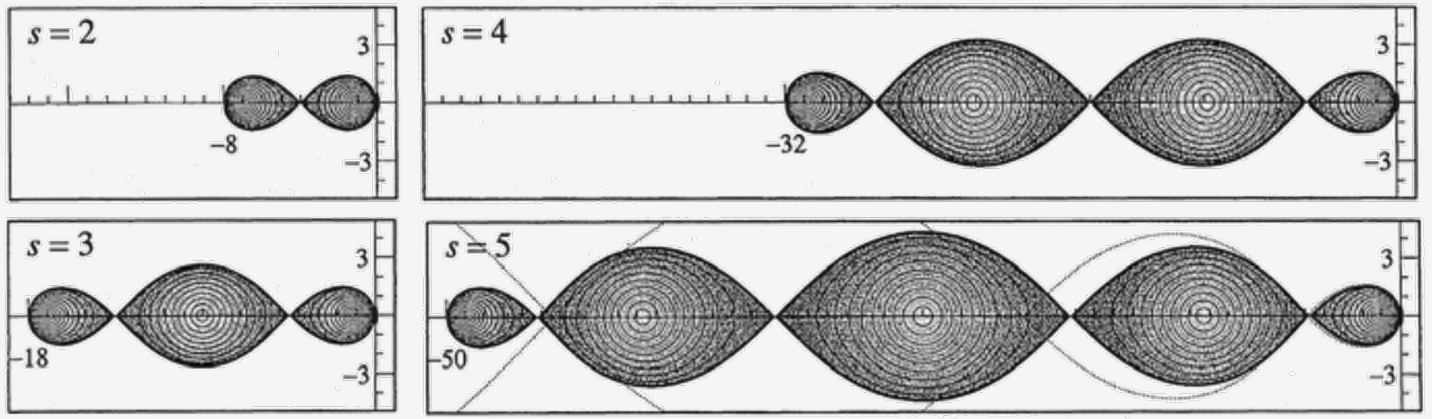


Fig. 2.12. Stability domains for shifted Chebyshev polynomials ($s = 2, 3, 4, 5$)
(dots represent limiting case $s \to \infty$, see Exercise 8 below)

（引用：Hairer–Wanner: Solving Ordinary Differential Equations II）

その他の代表的な数値解法クラス

一段解法：分離型RK法

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}u(t)\\ v(t)\end{bmatrix}=\begin{bmatrix}f_1(u(t),v(t))\\ f_2(u(t),v(t))\end{bmatrix}$$ のように2種類の変数について書かれた常微分方程式系に対して

（例：u：位置，v：運動量）

$\dot{u}(t)=f_1(u(t),v(t))$ と $\dot{v}(t)=f_2(u(t),v(t))$ に異なるRK法を適用

f_1 ⬆ $(A,\boldsymbol{b})$　　f_2 ⬆ $(\tilde{A},\tilde{\boldsymbol{b}})$

分離型Runge–Kutta法

$$u_{n+1}=u_n+h\sum_{i=1}^{s}b_i f_1(U_i,V_i) \qquad v_{n+1}=u_n+h\sum_{i=1}^{s}\tilde{b}_i f_2(U_i,V_i)$$

$$U_i=u_n+h\sum_{j=1}^{s}a_{ij} f_1(U_j,V_j) \qquad V_i=u_n+h\sum_{j=1}^{s}\tilde{a}_{ij} f_2(U_j,V_j)$$

一段解法：splitting法

$\frac{\mathrm{d}}{\mathrm{d}t}u(t) = f_1\big(u(t)\big) + f_2\big(u(t)\big)$ のように右辺が2つ（以上）の項の和で表されていて

$\dot{u}(t) = f_1\big(u(t)\big)$ と $\dot{u}(t) = f_2\big(u(t)\big)$ が，それぞれ

- 元のODEよりも数値計算しやすかったり
- 厳密に時間発展を計算出来たりする場合　　に有用な数値解法

Splitting解法の例

$\Phi_h^1(u) : \dot{u}(t) = f_1\big(u(t)\big)$ の h-時間発展写像（あるいはその近似）

$\Phi_h^2(u) : \dot{u}(t) = f_2\big(u(t)\big)$ の h-時間発展写像（あるいはその近似）

$u_1 := \Phi_{\frac{h}{2}}^1 \circ \Phi_h^2 \circ \Phi_{\frac{h}{2}}^1(u)$ ⬅ Strang splitting

Splitting解法：こんな感じで写像の合成で近似解を定義する解法

一段解法：exponential integrators

ODEの線形部分は，そこだけなら大体の場合効率よく（厳密に）計算できる

例 $\dfrac{\mathrm{d}}{\mathrm{d}t}u(t) = Au(t) + g(u(t))$

厳密に計算

厳密解：$u(h) = \mathrm{e}^{hA}u_0 + \displaystyle\int_0^h \mathrm{e}^{(h-s)A} g\big(u(s)\big)\,\mathrm{d}s$

近似計算する

例えば **Exponential Euler 法**

$$u_1 = \mathrm{e}^{hA}u_0 + h\varphi_1(hA)g(u_0) \qquad \varphi_1(z) = \frac{\mathrm{e}^z - 1}{z}$$

➡ この考え方をRunge–Kutta法やsplitting解法と組み合わせる

Composition法（合成解法）

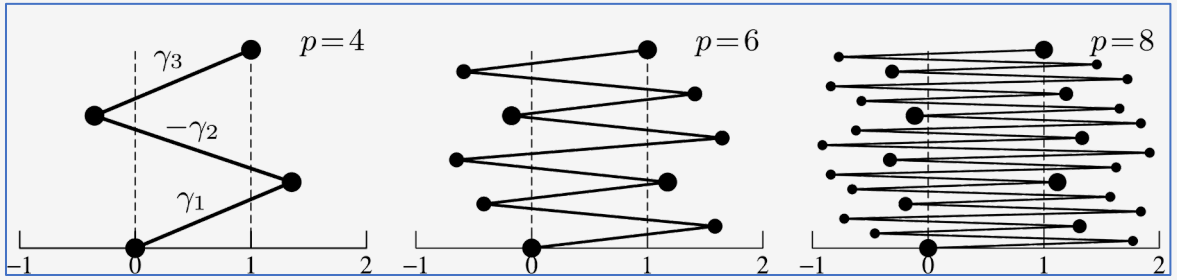
Composition ： 一段解法の精度を上げる技巧

$$u_1 = \Phi_{\gamma_s h} \circ \cdots \circ \Phi_{\gamma_1 h}(u_0) \qquad (\gamma_1 + \cdots + \gamma_s = 1)$$

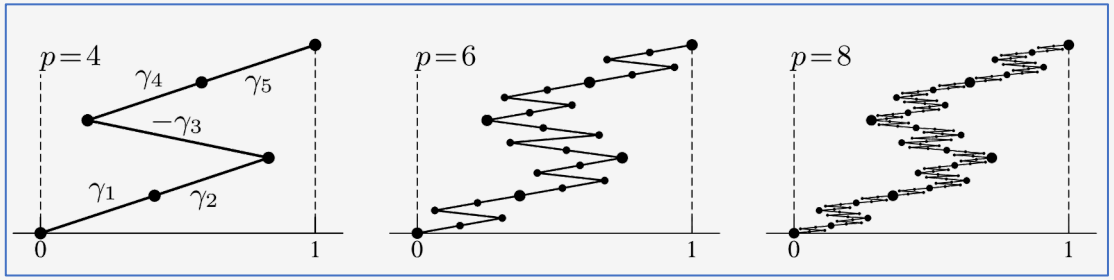
例えば $\gamma_1 = \cdots = \gamma_s = \frac{h}{s}$ とすれば数値計算の誤差は小さくなりそうだが…

進んで戻って進むと…次数が上がる！

YoshidaのTriple Jump



SuzukiのFractal



（引用：Hairer, Lubich, Wanner 2006）

Φ_hが対称な解法（刻み幅を$-h$にして計算すれば一つ前の近似解と一致する解法）のとき，次数が2ずつ向上する

多値法：線形多段解法 (Linear multistep methods)

$$\frac{\mathrm{d}}{\mathrm{d}t}u(t) = f\big(u(t)\big)$$

線形多段解法

$$u_n = a_1 u_{n-1} + \cdots + a_k u_{n-k} + hb_0 f(u_n) + hb_1 f(u_{n-1}) + \cdots + hb_k f(u_{n-k})$$

（例）2次のAdams–Bashforth法

$$u_n = u_{n-1} + \frac{3}{2}hf(u_{n-1}) - \frac{1}{2}hf(u_{n-2}) \quad \Leftrightarrow \quad \frac{u_n - n_{n-1}}{h} = \frac{3}{2}f(u_{n-1}) - \frac{1}{2}f(u_{n-2})$$

陽的Euler法と同程度のコストで2次を達成！

一般線形法 (General linear methods)

RK法 + 線形多段解法

線形多段解法の精度と安定性の関係

精度と安定性の定義は省略するが…

第1のDahlquistバリア

安定な p 次精度の k 段法について以下が成立

- $p \leq k+2$ (k: 偶数)
- $p \leq k+1$ (k: 奇数)
- $p \leq k$ ($\frac{\beta_k}{\alpha_k} \leq 0$，特に陽解法)

第2のDahlquistバリア

陽的な線形多段解法はA安定ではない

構造保存数値解法
（幾何学的数値解法）

Störmer法

Newtonの運動方程式 $\frac{\mathrm{d}^2}{\mathrm{d}t^2}q = -\nabla V(q)$ $\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}q\\p\end{bmatrix} = \begin{bmatrix}p\\-\nabla V(q)\end{bmatrix}$

Störmer 法 (1907)

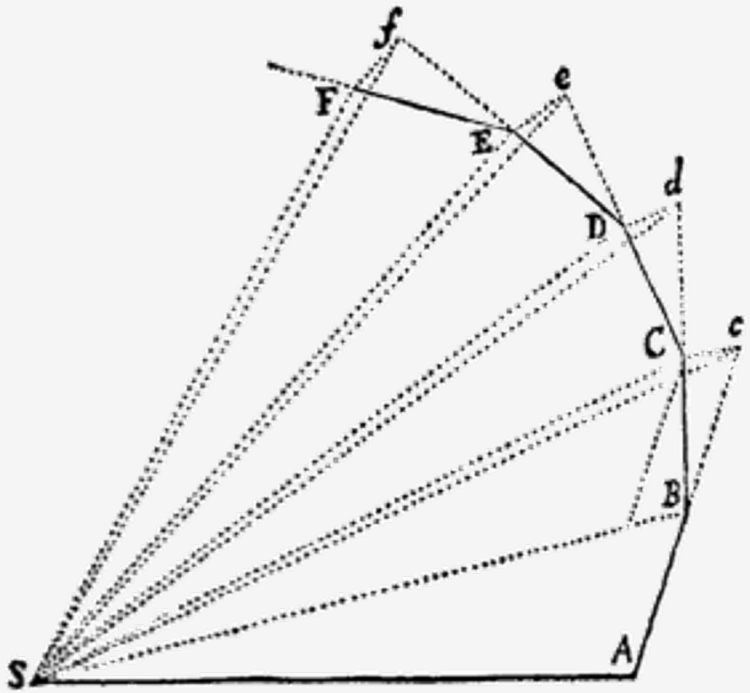
$$q_{n+1/2} = q_n + \frac{h}{2}p_n$$ 位置 q を 陽的Euler法 で半ステップ進める

$$p_{n+1} = p_n - h\nabla V(q_{n+1/2})$$ 運動量 p を 1ステップ進める

$$q_{n+1} = q_{n+1/2} + \frac{h}{2}p_{n+1}$$ 位置 q を 残りの半ステップ進める

- **Verlet 法** (1967; 分子動力学) も本質的に同じ解法
- 数値解析学者は **Störmer–Verlet 法**と呼んだりする

NewtonのPrincipia



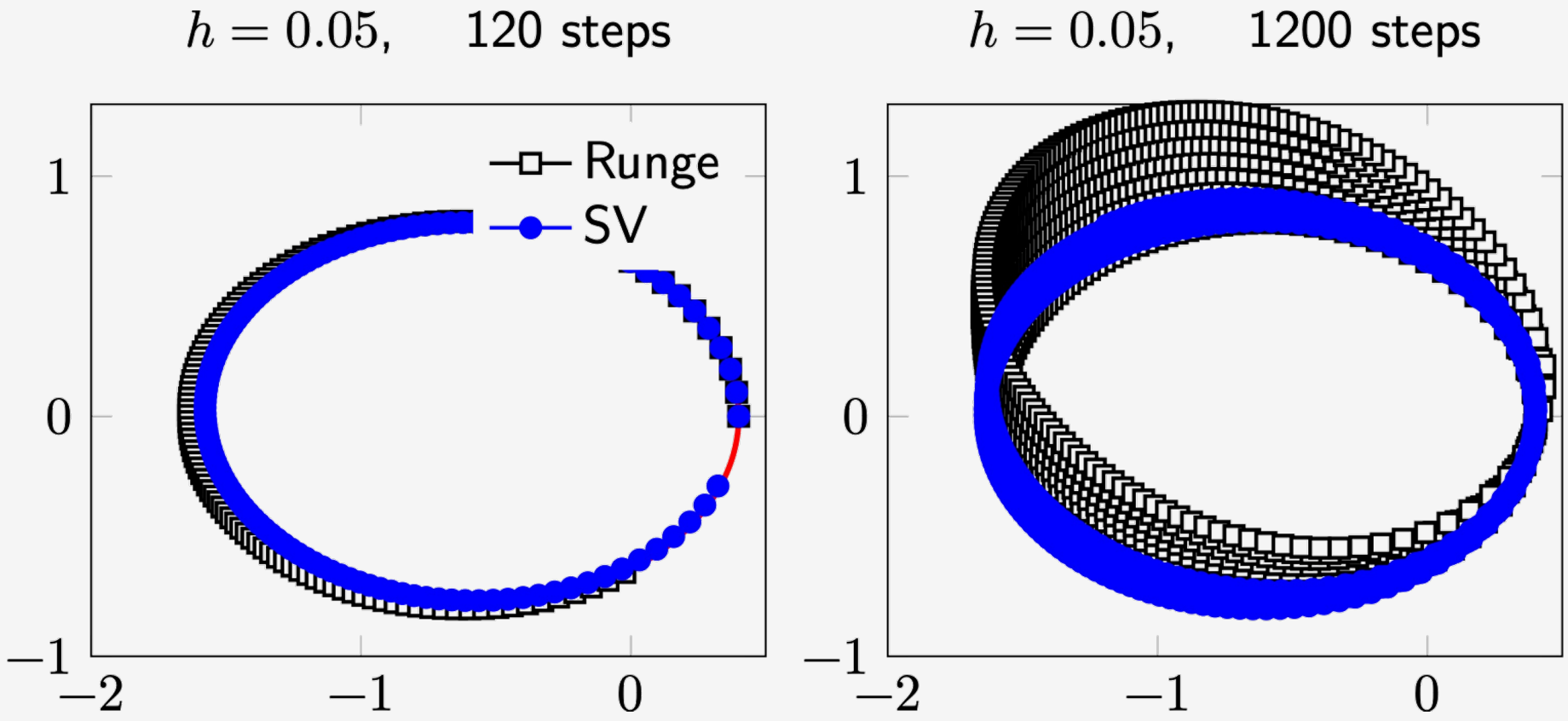
（引用：Newton, Principia）

数値計算というよりは，**Keplerの第2法則**（角運動量保存則）を
説明するためにさきほどの公式が用いられている

Newton–Störmer–Verlet 法 と呼ぶことも（まれに）ある

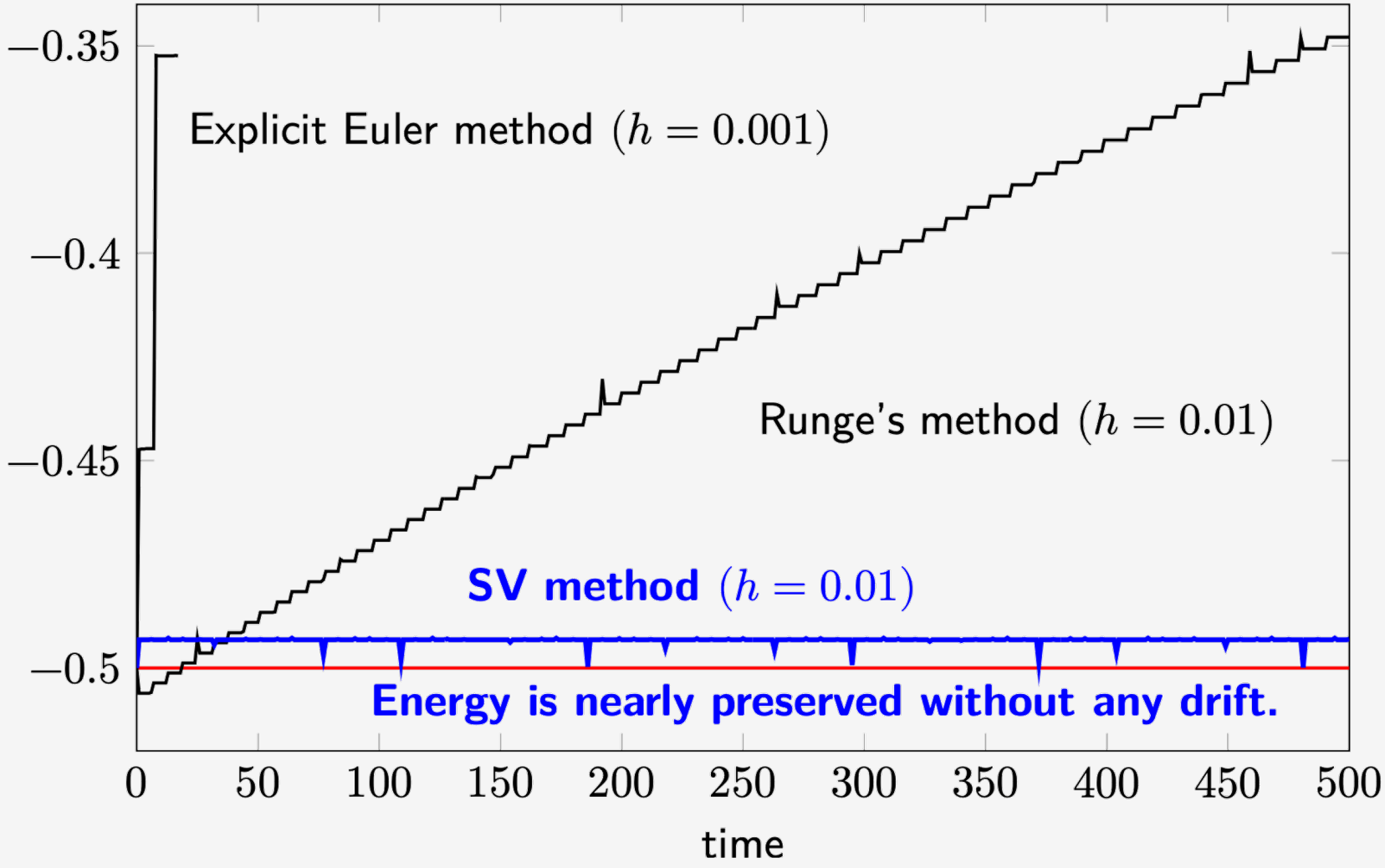
Keplerの2体問題の数値計算

「Störmer–Verlet 法」と「Rungeの公式」を比較（共に2次精度解法）



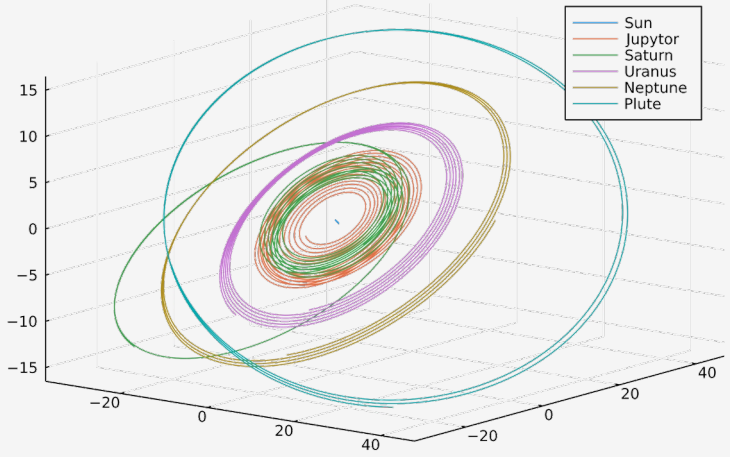
計算する時間が長くなるとSV方が有利

エネルギー保存則はどうなっている？

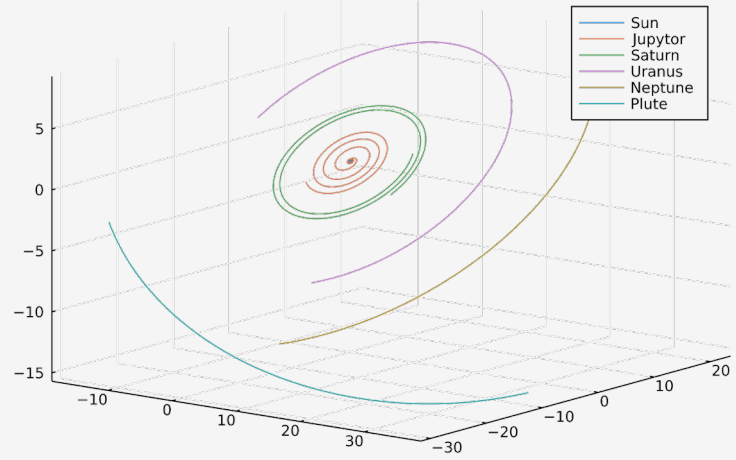


太陽系のシミュレーション

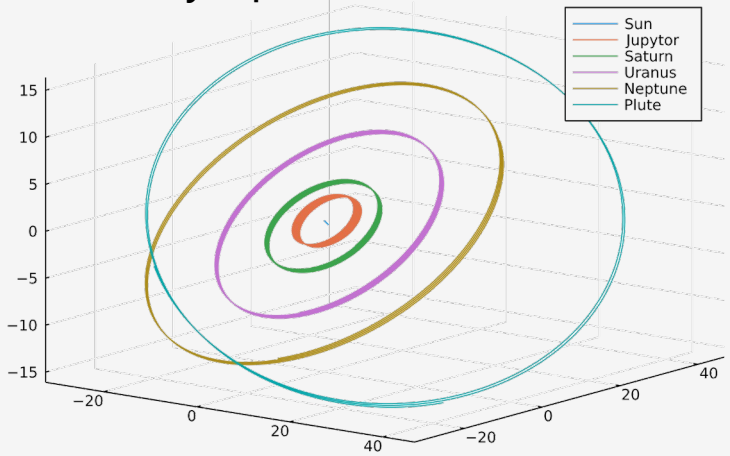
陽的Euler 法



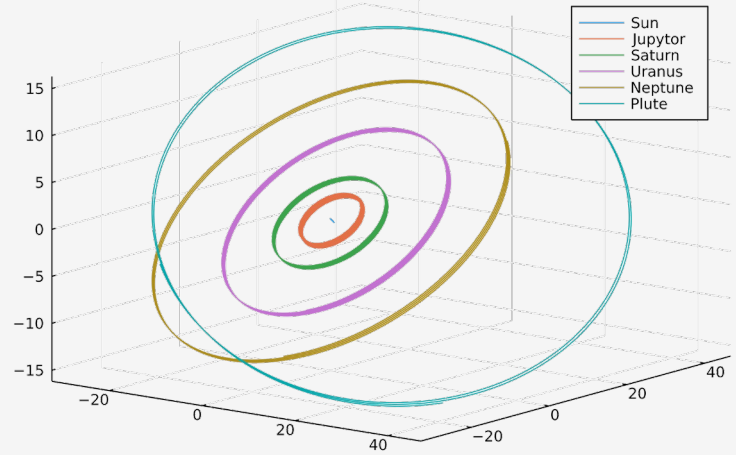
陰的Euler 法



Symplectic Euler 法



Störmer–Verlet 法



（提供：岩瀬さん）

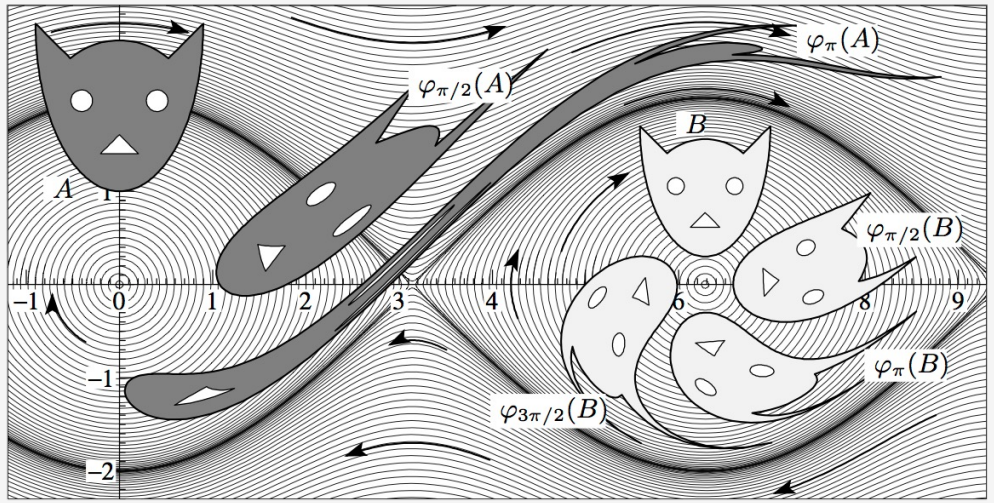
Störmer–Verlet 法はなぜ優れている？

Newtonの運動方程式は Hamilton系

解の時間発展写像はシンプレクティック

SV法による近似解の離散的な時間発展写像もシンプレクティック

もとの系に非常によく似たHamilton系を厳密に解いている



$\mathbb{R}^2$ の場合，

シンプレクティック性

＝ 位相空間内の面積保存

（引用：Hairer, Lubich, Wanner 2006）

幾何学的数値積分法・構造保存数値解法

問題の数理的構造を離散化後も厳密に再現する数値解法

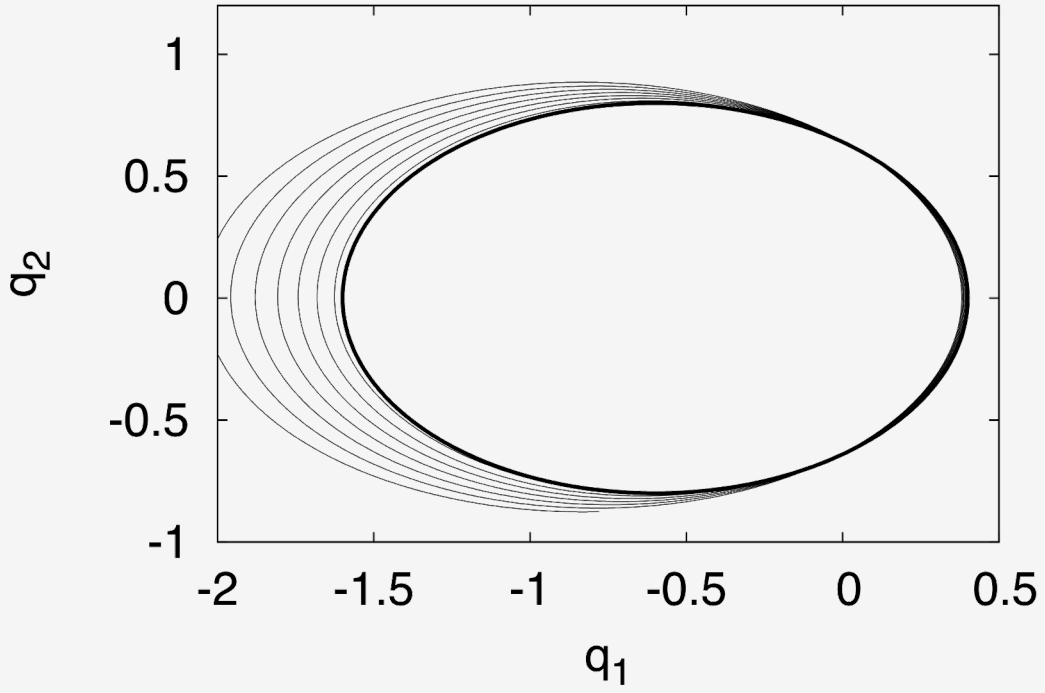
- シンプレクティック解法
- エネルギー保存解法（シンプレクティック性とは両立しない）
- エネルギー散逸解法
- 多様体上のODEに対する数値解法
- 高振動系やマルチスケール問題に対する数値解法

研究者の主なタスク

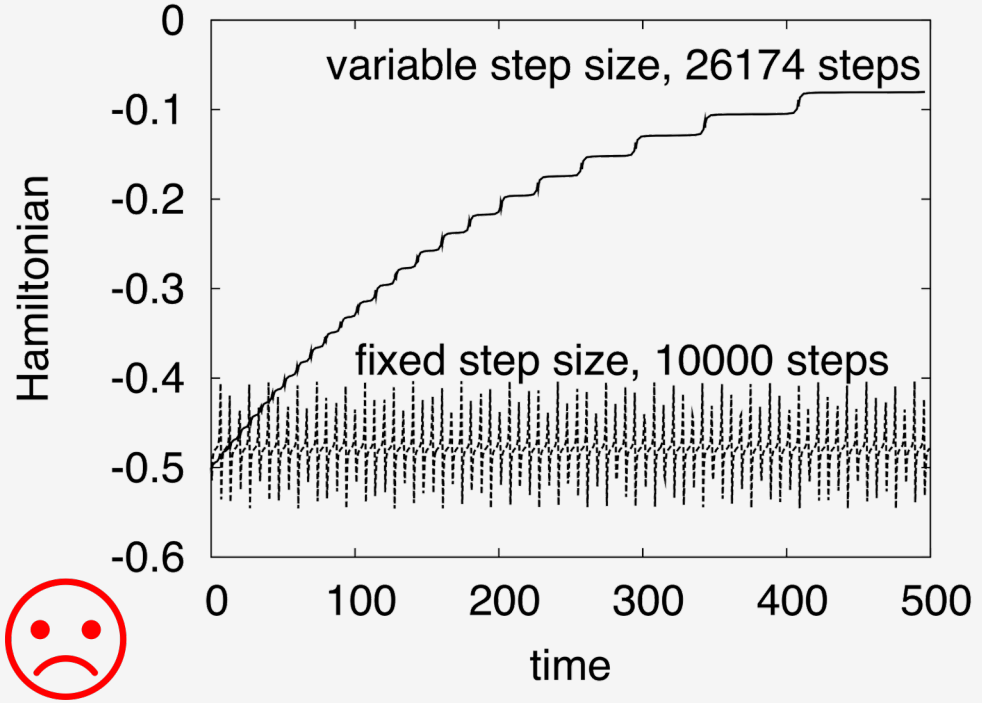
- どのように構造保存アルゴリズムを設計すればよいか？
- 恩恵は何か？

DANGER 刻み幅制御と組み合わせてみよう

Kepler問題に対して，symplectic Euler法に刻み幅制御を組み込んでみる



外側に膨れている…

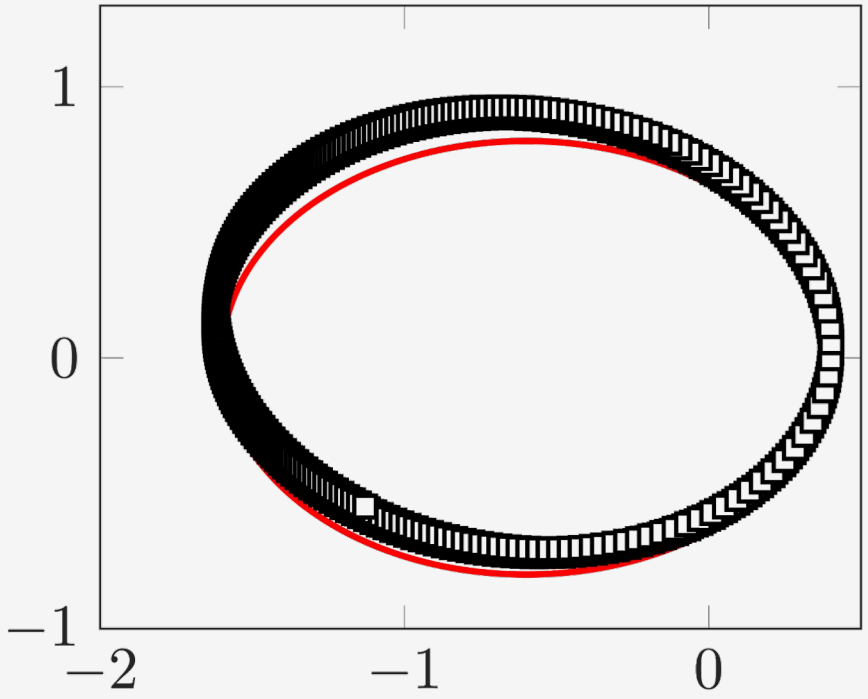


☹

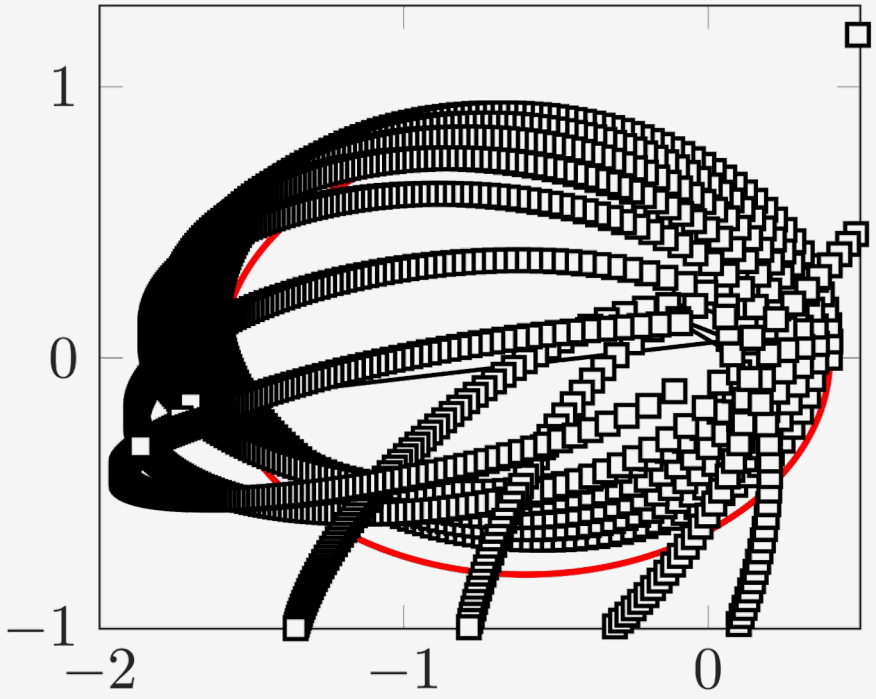
ステップサイズは小さくなっているのに，
振る舞いは悪化…

DANGER シンプレクティック + エネルギー保存

Symplectic Euler法



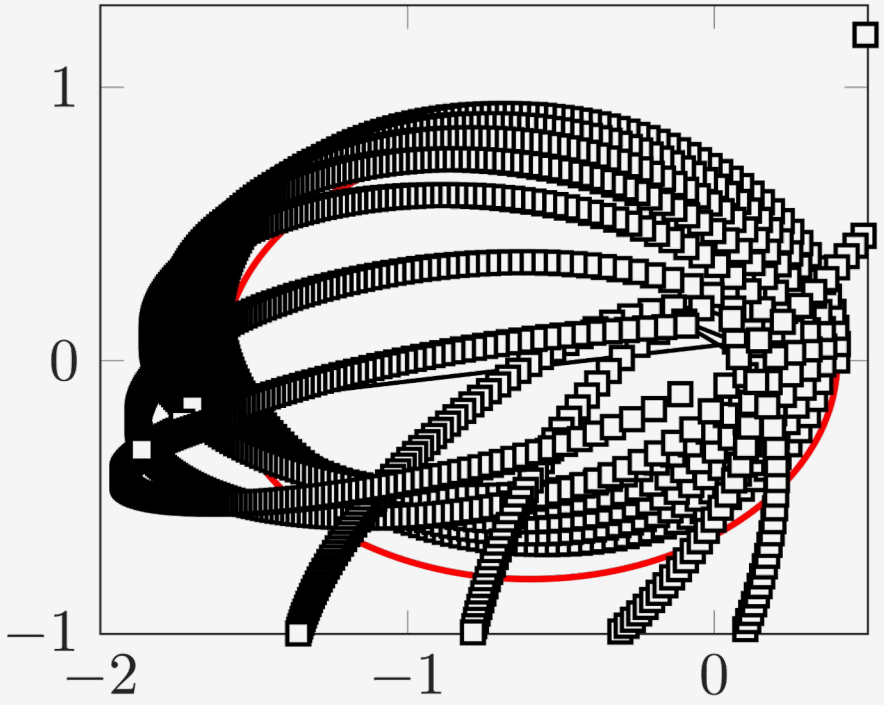
Symplectic Euler法
+
各ステップでエネルギーが保存するように射影



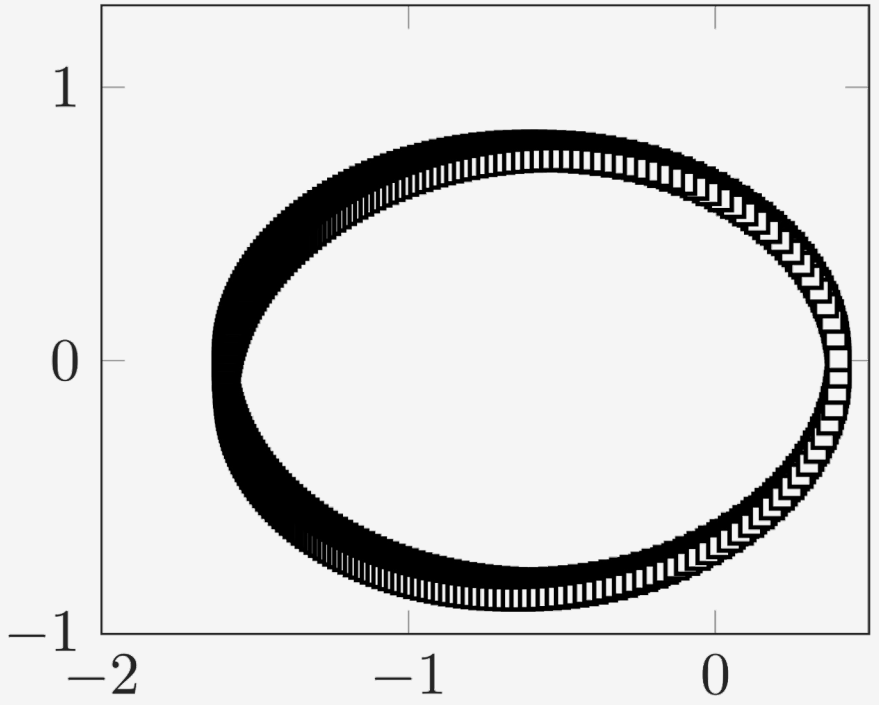
強引に組み合わせると，元よりも悪化しうる

DANGER シンプレクティック + エネルギー保存

Symplectic Euler法
+
各ステップでエネルギーが保存するように射影



Average vector field 法
(Quispel–McLaren, 2008)



適切なエネルギー保存解法だと良好な結果が得られる

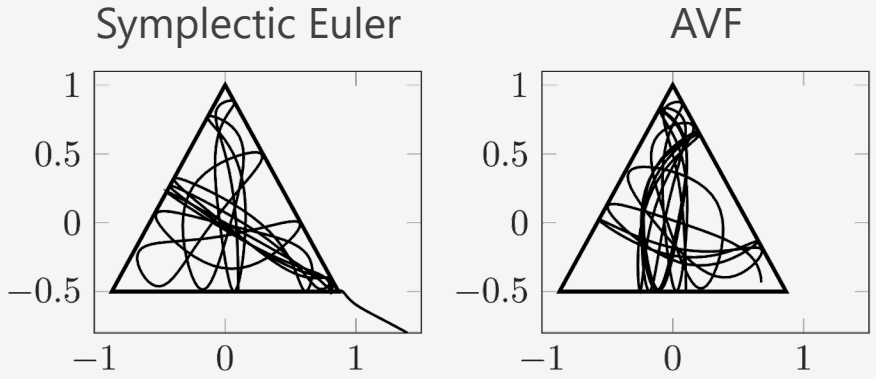
シンプレクティック vs エネルギー保存

シンプレクティック解法 と エネルギー保存解法 のどちらがよいか？

- Hamilton系に対してはシンプレクティック解法が好ましいことが多い
  （陽的な解法も知られている）
- しかし，基本的にはケース・バイ・ケース
- エネルギー保存解法は散逸系に対する散逸解法とも密接な関連

Symplectic解法で不安定になる例もある

例：Hénon–Heiles系



教訓

良かれと思って 良いもの同士を組み合わせると，大変なことになることがある

- あれ？と思ったら，マシンパワーに頼る前に
  そもそもの恩恵を（理由も含めて）しっかり理解することが肝心
- 大規模な問題ではなく，（特徴の近い）小規模な問題で考察するのも大事
- 必要に応じて数値解析の専門家にご相談ください

結局，数値解法はどう選択すればよい？

個人的な意見ですが，だいたい同意してくださる方も多いのではと思います

- 汎用解法を使うなら，どれが良さそうかを考える
  - 問題なく動くならその選択で十分！
  - とはいえ，汎用解法もたくさん種類がある
    - ➡ 常微分方程式の数値解法I, II を参照するのがおすすめ
- 不満が残るなら（あるいはもっと改善したいなら），
  ODEの構造をよく考察し，構造保存数値解法の適用を考えてみる
  - 幾つかの技巧を組み合わせる際は注意が必要

ODEの並列計算

ODEの数値計算の時間並列化

通常，**並列計算** というと **空間方向の並列化**

スパコンが大活躍！

（スパコンについては後述）

そもそも時間方向の並列化は可能？

いってみれば，明日と明後日と一週間後の天気の予測を
順々にシミュレーションするのではなく同時に行うようなもの...

A. 完全に独立に同じ精度で...というのは困難だが

ある程度の並列化は可能！

例えば，部分的に並列計算を行うことで，
順々に計算するよりも最終的なコストを小さくすることを目指す

Parareal法

時間も限られているので，時間並列計算の手法の中でも利用頻度の高い
Parareal法 について，そのイメージをお話します．

Parareal法を含む時間並列計算の全体像を掴むには，
以下の会議論文がおすすめです：

M. Gander
50 Years of Time Parallel Time Integration, 2015.
https://doi.org/10.1007/978-3-319-23321-5_3

Parareal法

$$\frac{\mathrm{d}}{\mathrm{d}t}u(t) = f\big(u(t)\big), \quad u(0) = u_0$$

$\mathcal{F}_h$: Fine integrator ⬅ 高精度だが，これで時間全区間を逐次的に計算するのはコスト的に厳しい…

$\mathcal{G}_h$: Coarse integrator ⬅ 時間全区間を逐次的に計算するのは容易だが低精度

Parareal法 (Lions, Maday, Turinici 2021)

- $u_0^0 = u_0$ and $u_{n+1}^0 = \mathcal{G}_h(u_n^0)$
- For $k = 1,2, \ldots$, $u_0^k = u_0$ and

$$u_{n+1}^{k+1} = \mathcal{G}_h\big(u_n^{k+1}\big) + \boxed{\mathcal{F}_h\big(u_n^k\big)} - \mathcal{G}_h\big(u_n^k\big)$$

$n = 1,2, \ldots$ に対して**並列に計算可能**

例題

$$\frac{\mathrm{d}}{\mathrm{d}t}u(t) = \begin{bmatrix} -1 & 6 \\ -6 & -1 \end{bmatrix} u(t), \qquad u(0) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$\mathcal{F}_h$: Fine integrator

陰的Euler：刻み幅 $h = 0.1/16$ で 16ステップ

$\mathcal{G}_h$: Coarse integrator

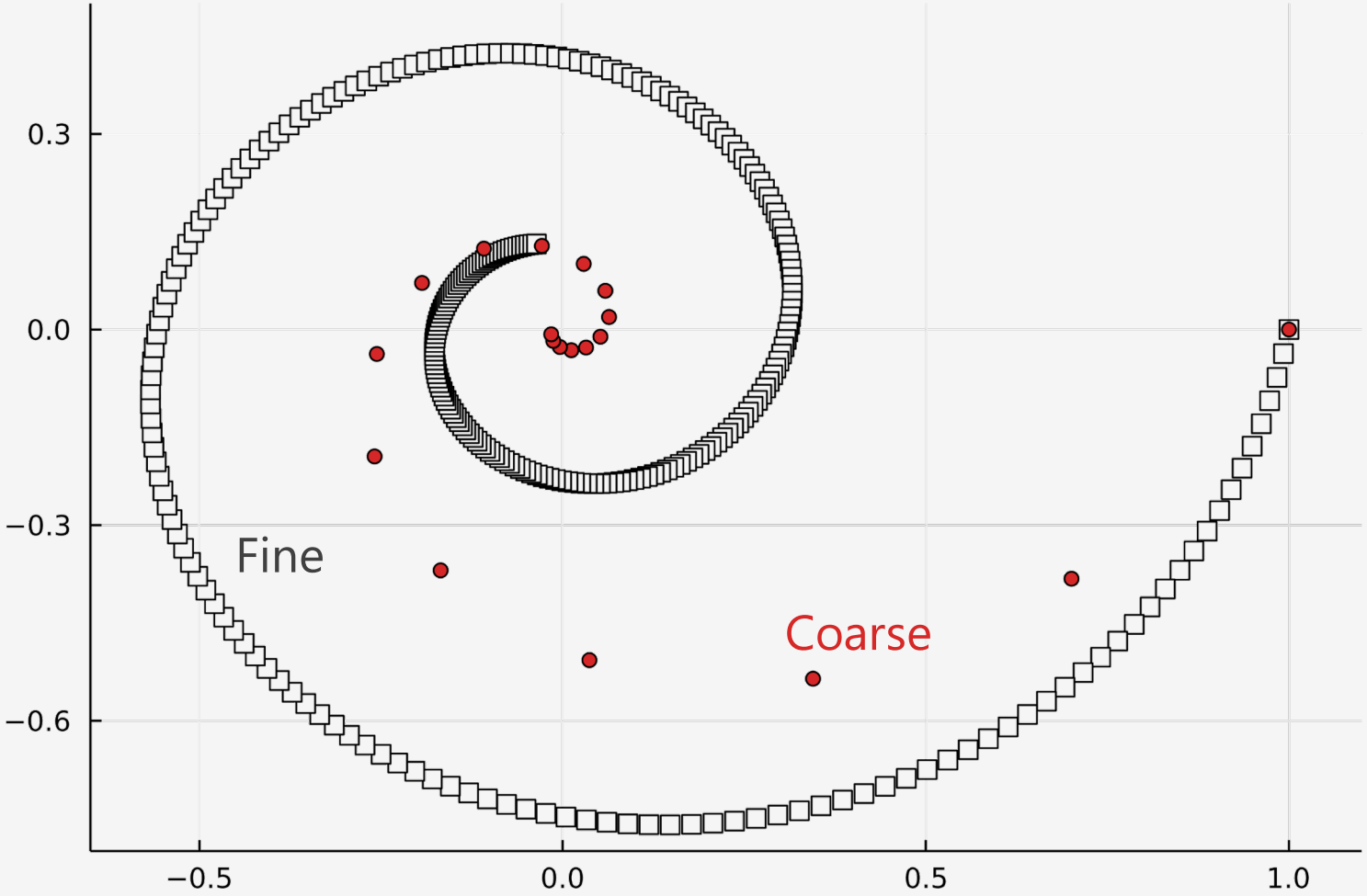
陰的Euler：刻み幅 $h = 0.1$

（Coarseと同じ解法で刻み幅がとても小さい解法

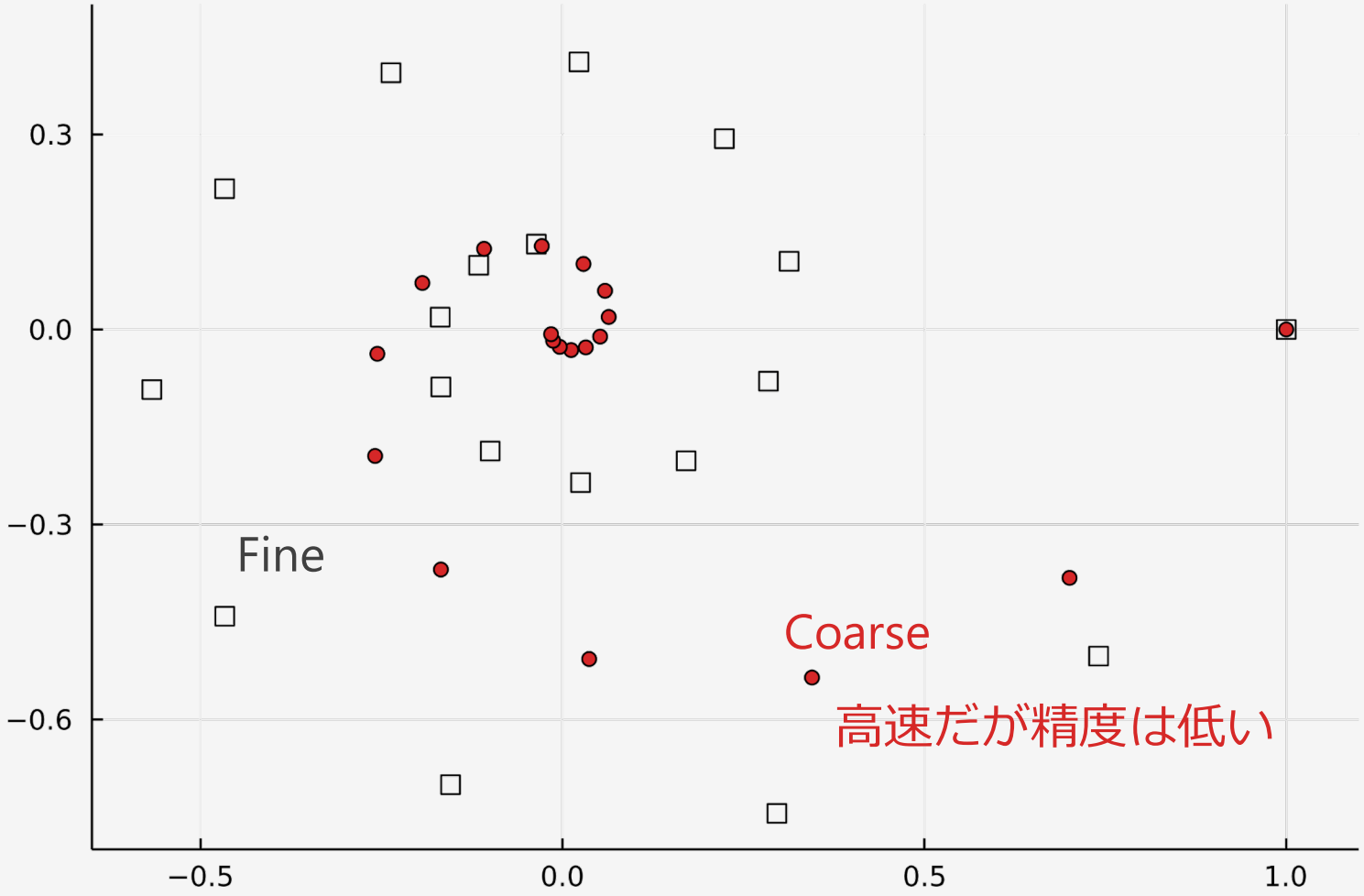
or Coarseよりも次数が高い解法

or 次数は高く刻み幅は小さい解法）

Fine integrator と Coarse integrator

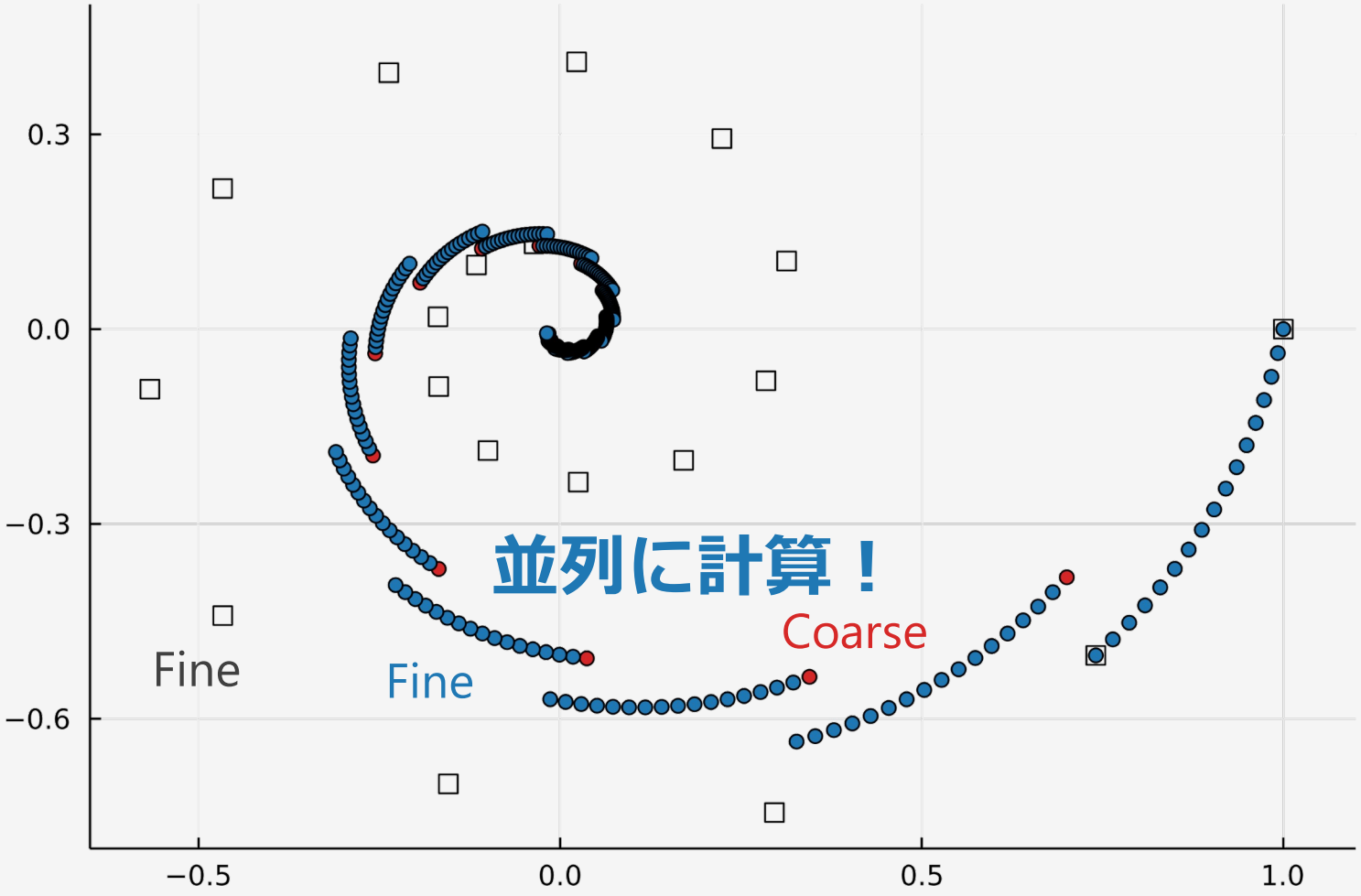


Fine integrator と Coarse integrator

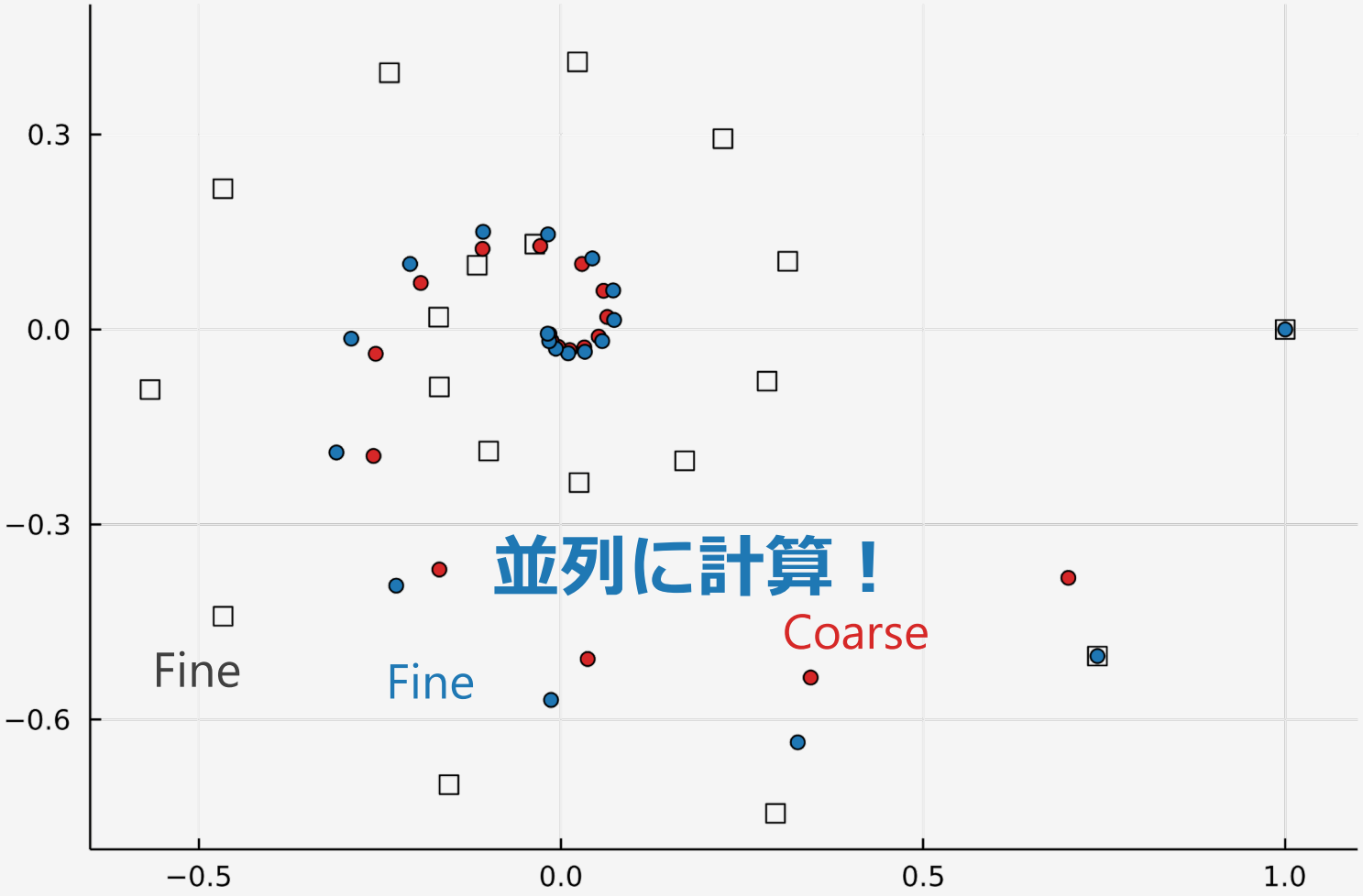


Fineを直接計算するよりも効率よくFineに近い結果を得たい！

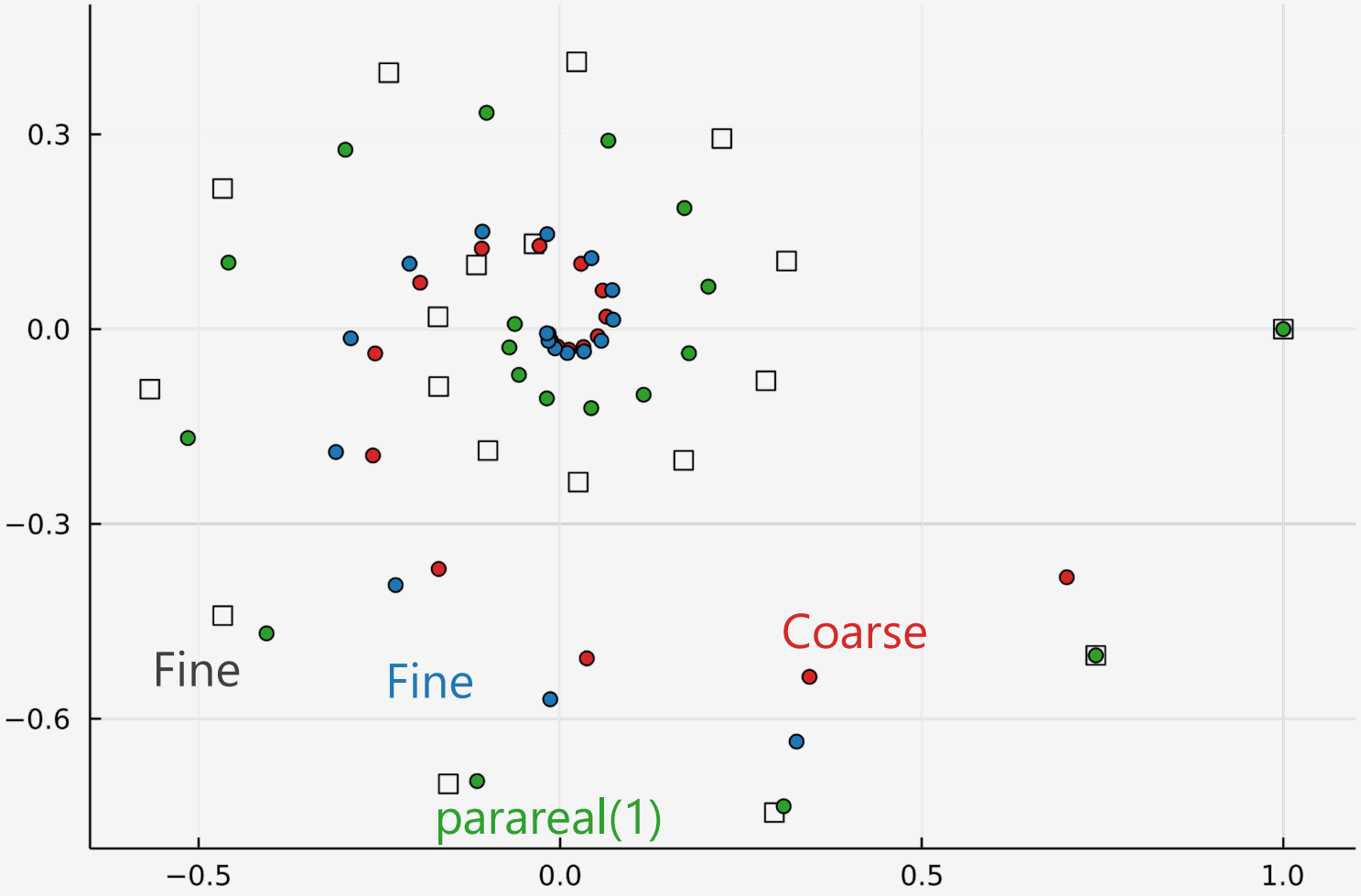
Parareal 1反復目



Parareal 1反復目

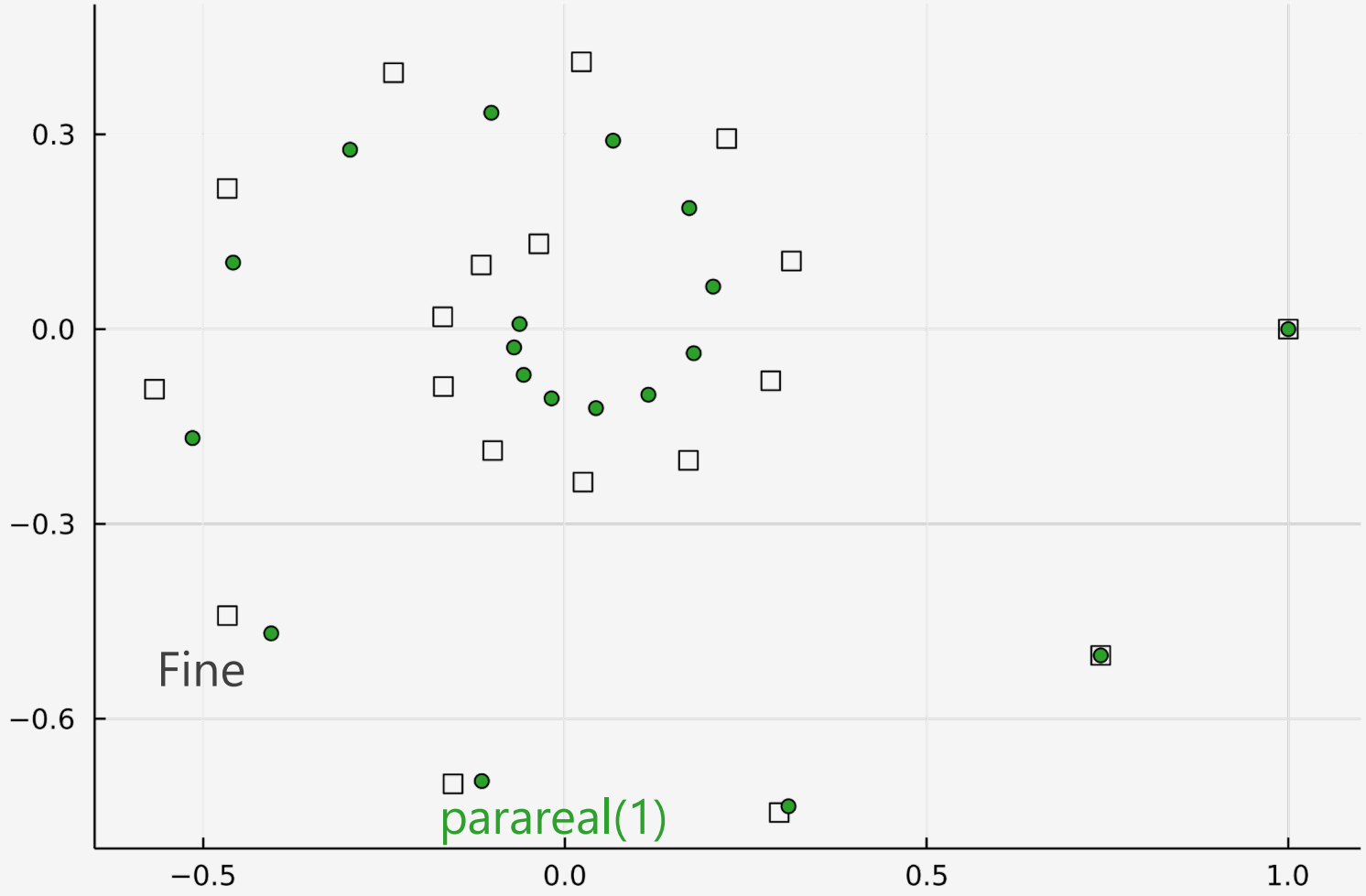


Parareal 1反復目

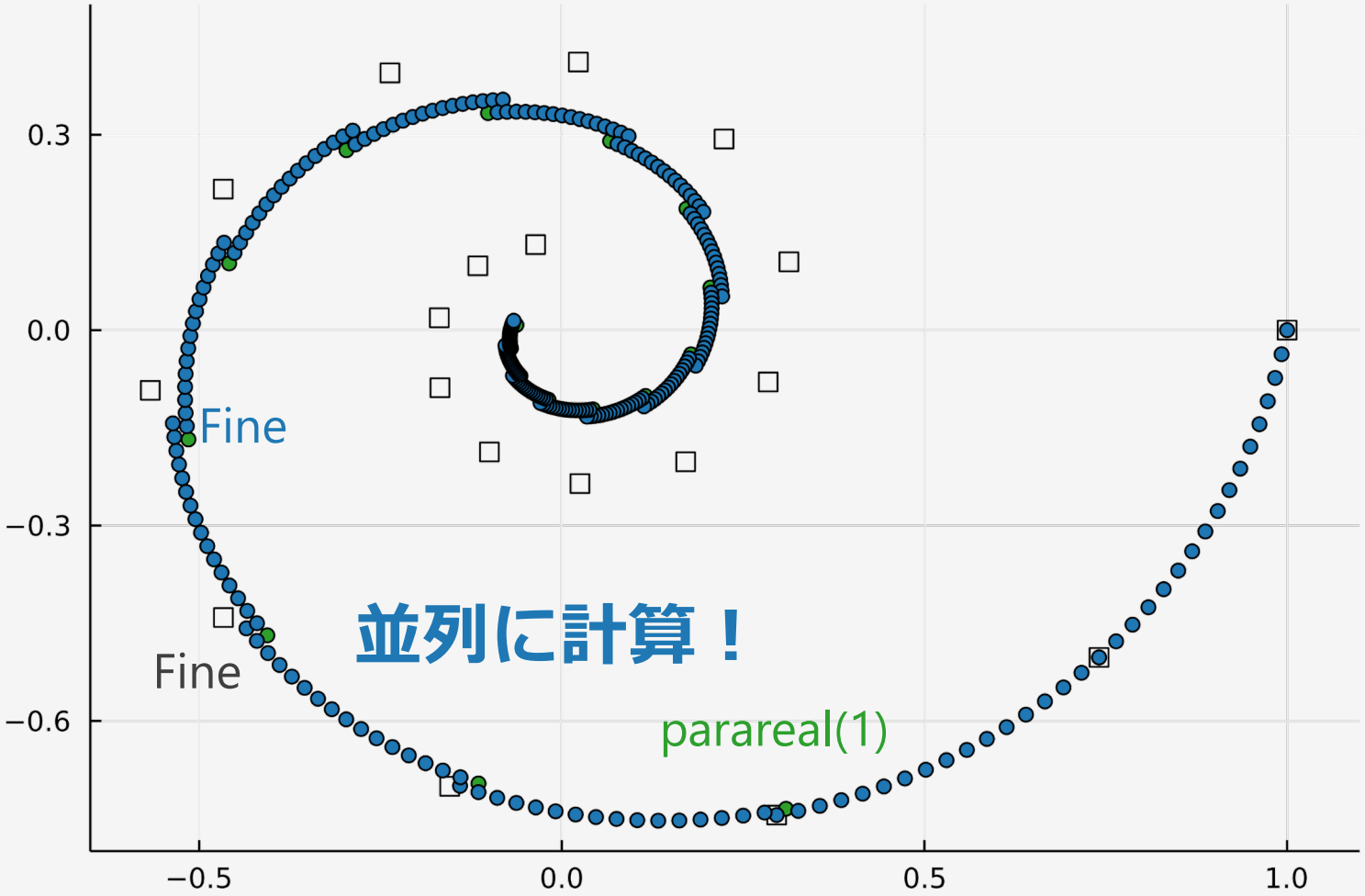


Coarseで最後まで時間発展するくらいのコスト

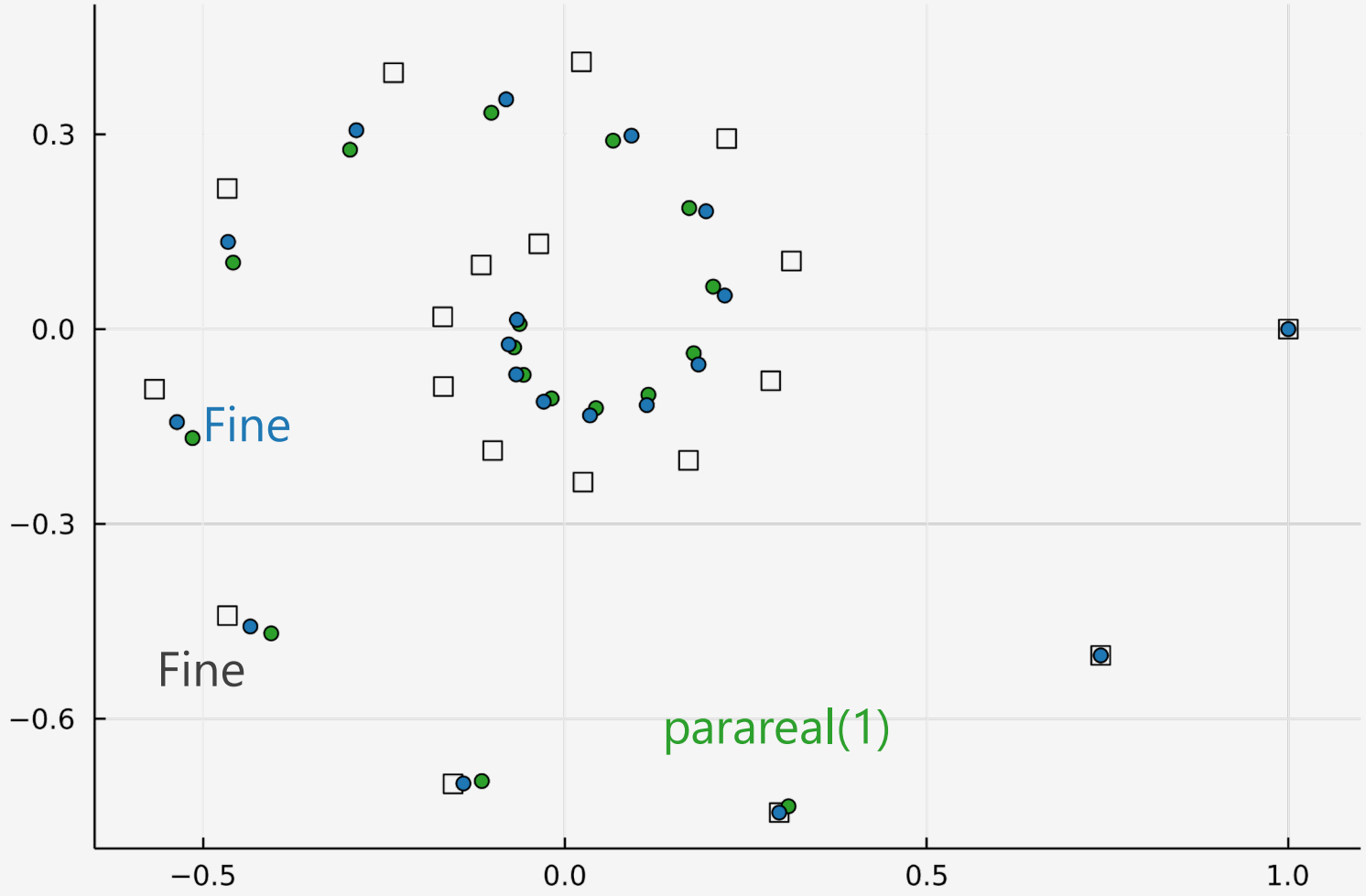
Parareal 1反復目



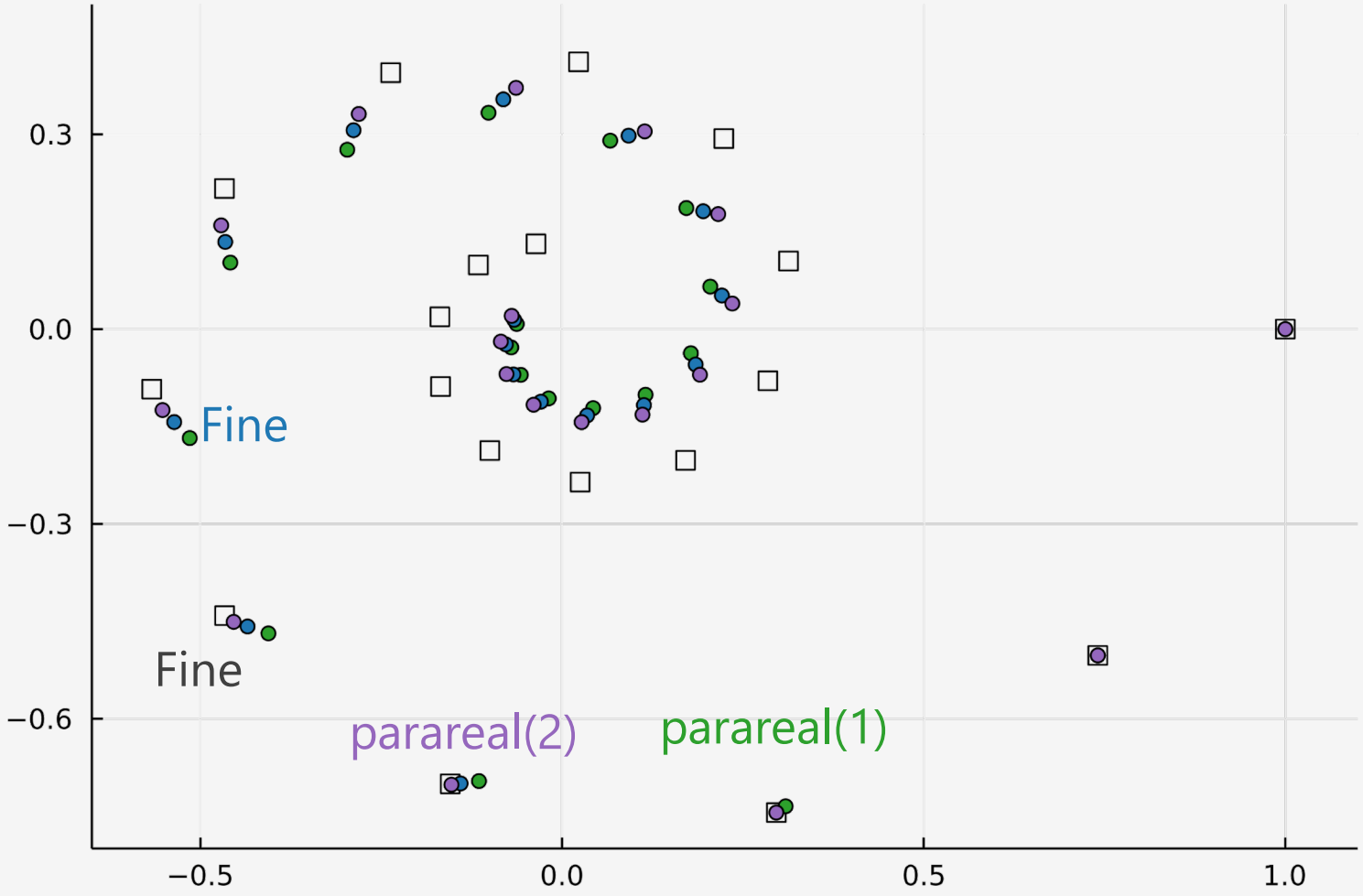
Parareal 2反復目



Parareal 2反復目

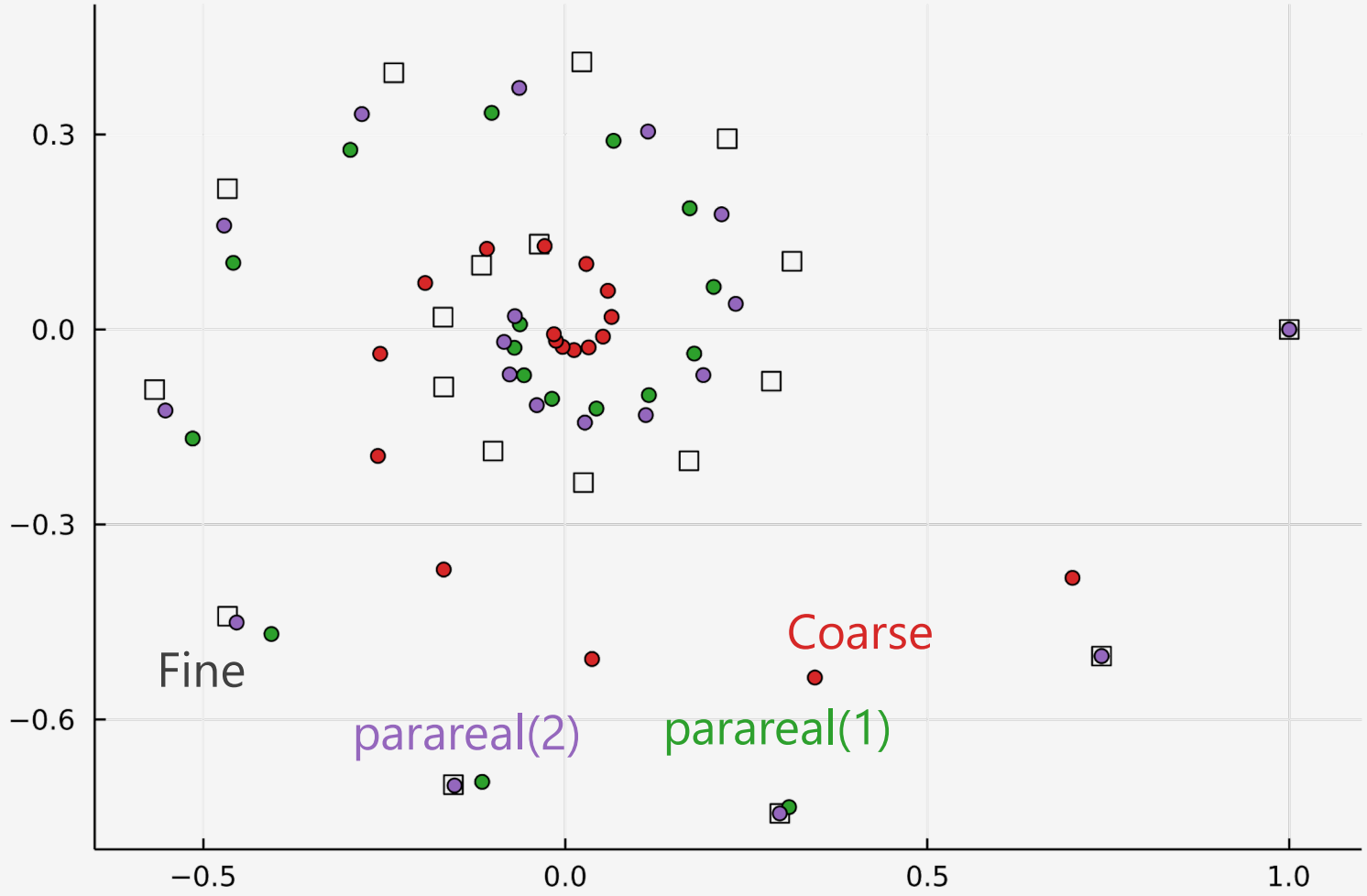


Parareal 2反復目

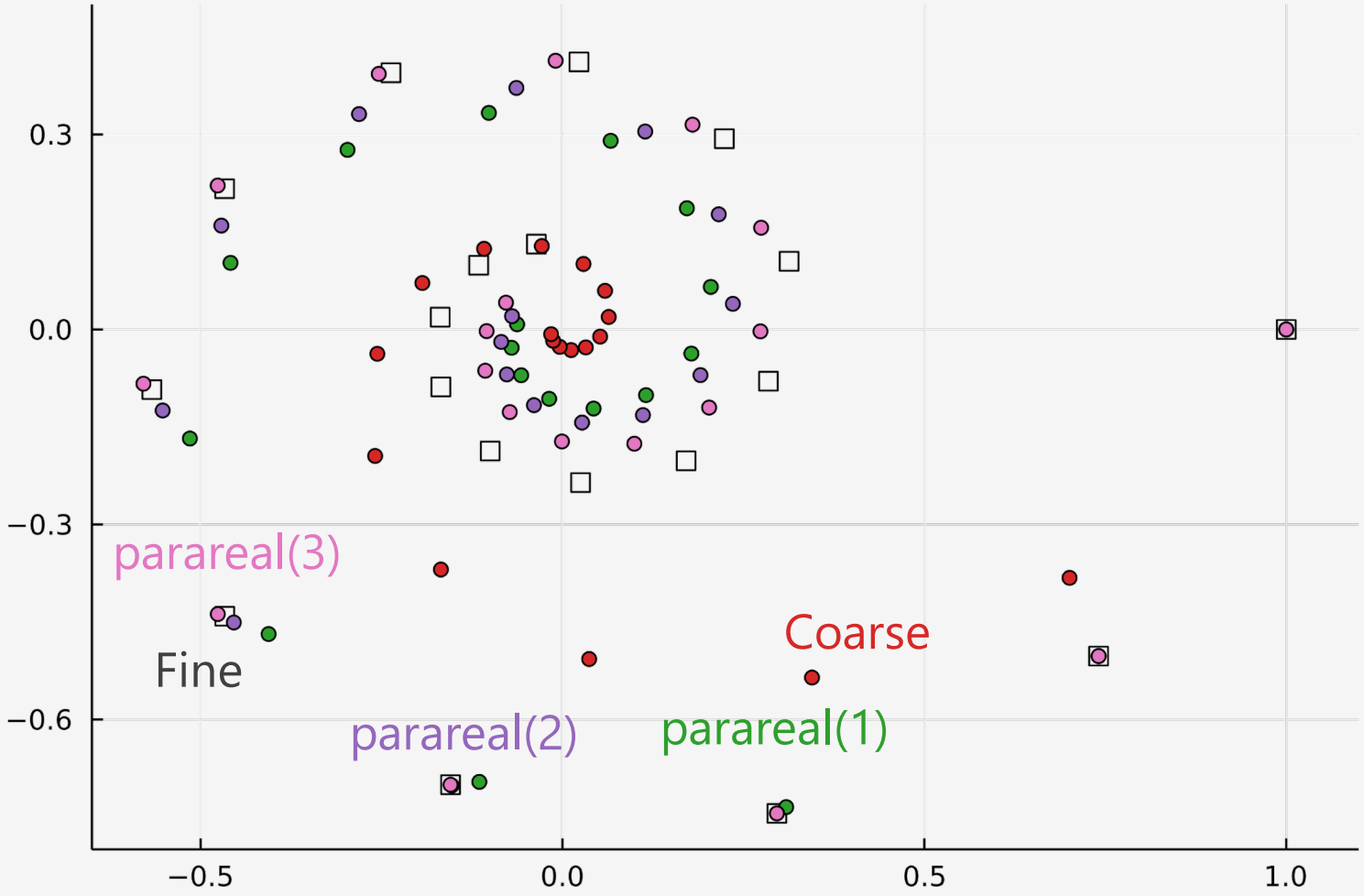


Coarseで最後まで時間発展するくらいのコスト

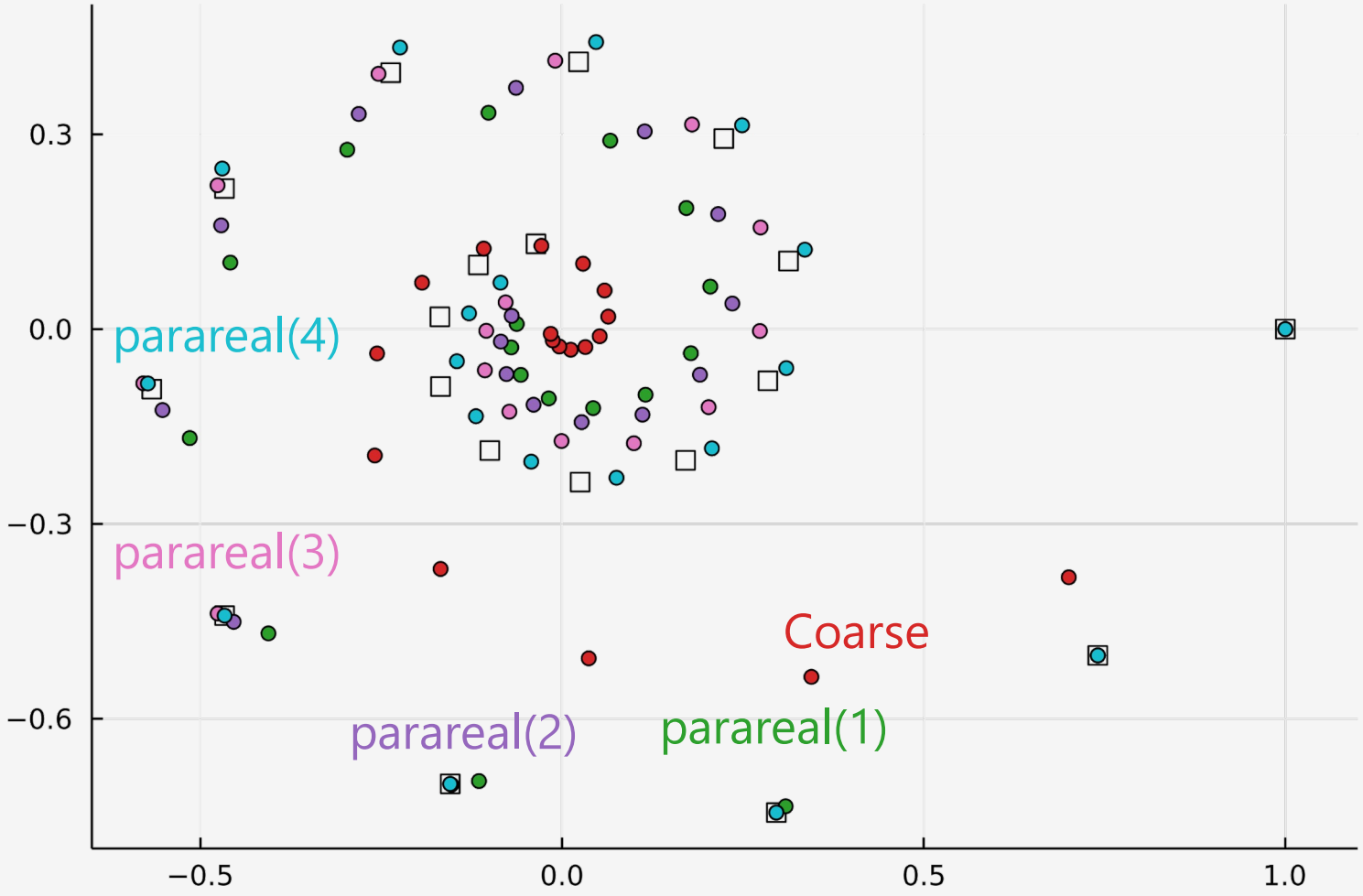
Parareal 2反復目



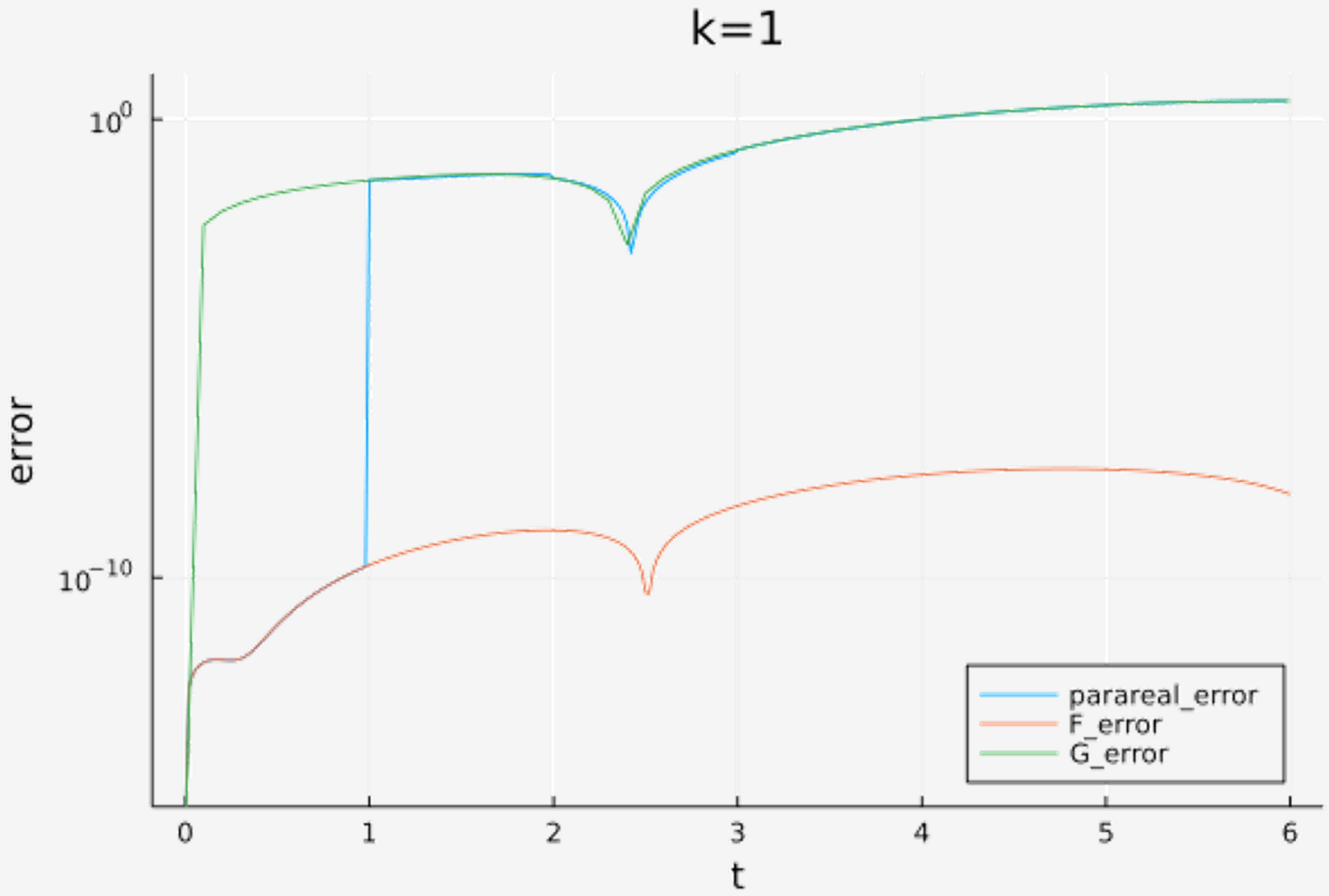
Parareal 3反復目



Parareal 4反復目



Kepler問題に対する収束の様子



（提供：縄手さん）

阪大のスパコン（特にSQUIDについて）

- 阪大のスパコンの１つ
- Top500で現在104位
- 毎年６月頃と9月頃に様々な講習会が開催されている

  http://www.hpc.cmc.osaka-u.ac.jp/lecture_event/
- 利用方法：http://www.hpc.cmc.osaka-u.ac.jp/squid/

システム構成

- 1ノード76コア
- 小規模な問題の時間並列のテストなら
  複数ノードまでは必要ない
- １週間で試してみるにはちょうどよいかも
- ちょっとしたテストならフロントエンドで
  Jupyter Notebookも使える
  （PythonやJulia, Rも使える）

総演算性能	16.591 PFLOPS		
ノード構成	汎用CPUノード群 1,520 ノード(8.871 PFLOPS)	プロセッサ：Intel Xeon Platinum 8368 (Icelake / 2.40 GHz 38コア) 2基 主記憶容量：256GB	
	GPUノード群 42 ノード(6.797 PFLOPS)	プロセッサ：Intel Xeon Platinum 8368 (Icelake / 2.40 GHz 38コア) 2基 主記憶容量：512GB GPU：NVIDIA A100 8基	
	ベクトルノード群 36 ノード(0.922 PFLOPS)	ベクトルホスト (Vector Host)	AMD EPYC 7402P (2.8 GHz 24コア) 1基 主記憶容量：128GB
		ベクトル エンジン (Vector Engine)	NEC SX-Aurora TSUBASA Type20A(10コア) 8基 主記憶容量：48GB
ストレージ	DDN EXAScaler (Lustre)	HDD：20.0 PB NVMe：1.2 PB	
ノード間接続	Mellanox InfiniBand HDR (200 Gbps)		