電 301 数値解析
第 11 回
微分方程式の
数値解法 (1)

今回および次回の講義の参考文献 (1)

- 森正武, 数値解析, 第 2 版, 共立出版, 2002.
- 三井, 常微分方程式の数値解法, 岩波書店, 2003.
- 山本哲朗, 数値解析入門 [増補版], サイエンス社, 2003.
- 齊藤宣一, 数値解析入門, 東京大学出版会, 2012.
- 伊理正夫, 藤野和建, 数値計算の常識, 共立出版, 1985.

今回および次回の講義の参考文献 (2)

- E. Hairer, S. P. Nørsett and G. Wanner, Solving Ordinary Differential Equations 1, 2/e, Springer, 1993.
- E. Hairer, S. P. Nørsett and G. Wanner, Solving Ordinary Differential Equations 2, 2/e, Springer, 1996.
- J. C. Butcher, Numerical Methods for Ordinary Differential Equations, Wiley, 2003.
- R. Riaza, DIfferential-Algebraic Systems, World Scientific, 2008.

微分方程式とは (1)

- 未知関数とその (偏) 導関数のあいだに関係式が与えられたとき, これを微分方程式という (岩波数学入門辞典). 未知関数が 1 変数関数のとき常微分方程式, 多変数関数のとき偏微分方程式という (岩波数学入門辞典).
- 上記に関係式を満たす関数を求めることを, 微分方程式を解くという (同上).

微分方程式の数値解法の必要性 (1)

- 物理現象は微分方程式で表現できる.
- たとえば人工衛星を静止軌道に打ち上げるなどといった応用問題を解くためには, ロケットの挙動の精密な数学モデル (微分方程式モデル) を作る必要がある.

微分方程式の数値解法の必要性 (2)

- ロケットを静止軌道に打ち上げるためには推進装置や姿勢制御装置を適切に制御する必要がある．この制御の方法を試行錯誤によって決定しようとすると，失敗のたびにロケットが失われるから，危険でもあるし，費用がいくらあっても足りない．

微分方程式の数値解法の必要性 (3)

- ロケットの数学モデルに基づいて数値シミュレー ションをおこない, うまくいきそうな制御方式を決定してから実機を用いた実験をおこなうことにすれば, 失敗の可能性は劇的に低下する.
- このようなことをおこなうためには, 微分方程式を解く必要がある.

微分方程式の数値解法の必要性 (4)

- 見方を変えると, 物理現象は微分方程式をリアルタイムで解いているようなものであり, そのシミュレ– ションは, 微分方程式をコンピュータで解くことで物理現象の再現を試みるもの, と考えることもできる.
- 微分方程式は解析的に解けないことが一般的なので, 数値的な解法が必要になる.

微分方程式の数値解法の必要性 (5)

- 第 11 回と 12 回では, 常微分方程式の数値解法を取り扱う.
- 第 13 回と 14 回では, 偏微分方程式の数値解法を取り扱う.
- 今回の講義では, 常微分方程式 (とその一般化) に絞って議論を進める.

常微分方程式の分類 (1)

- 以下では, 未知関数を x とし独立変数 (実数) を t とした常微分方程式を考える.
- 独立変数の物理的意味は何でもよいが, 直感的には t が時間の場合が理解しやすいので, 以下では t を時間あるいは時刻と呼ぶ. とはいっても, 以下の議論は t の物理的意味には依存しない.

常微分方程式の分類 (2)

- 微分方程式にパラメータが含まれることもある.
- パラメータは時間に依存しない定数のこともあるが …
- 入力があるシステムの挙動が微分方程式で表現されているとき, 入力はこの微分方程式のパラメータとして取り扱われる. このような場合には, パラメータは時間の関数である.

常微分方程式の分類 (3)

- 以下しばらくパラメータを含まない微分方程式を考える. 未知関数はスカラーでもベクトルでもよい.
- 独立変数を t とし, 未知関数を $x(t)$ あるいは $\boldsymbol{x}(t)$ とする (未知関数がベクトル値のとき太字にする).

常微分方程式の分類 (4)

- 独立変数を x とし, 未知関数を $y(x)$ としている教科書も多いので注意すること.
- 未知関数の 1 階までの微分を含む常微分方程式 (系) の一般形は次の通り. ただし, n を未知関数の次元, m を式の数とする. 当面, $m = n$ であることは仮定しない.

常微分方程式の分類 (5)

n	m	微分方程式 (系)
1	1	$g\left(t, x, \frac{dx}{dt}\right) = 0$
≥ 2	1	$g\left(t, \boldsymbol{x}, \frac{d\boldsymbol{x}}{dt}\right) = 0$
1	≥ 2	$\boldsymbol{g}\left(t, x, \frac{dx}{dt}\right) = \mathbf{0}$
≥ 2	≥ 2	$\boldsymbol{g}\left(t, \boldsymbol{x}, \frac{d\boldsymbol{x}}{dt}\right) = \mathbf{0}$

x あるいは $\boldsymbol{x}$ は未知関数, $\mathbf{0}$ は零ベクトル

常微分方程式の分類 (6)

- 未知関数の n 階微分までを含む微分方程式の一般形 (ただし未知関数がスカラーで式が 1 個) は, 次のようになる.

$$h\left(t, x, \frac{dx}{dt}, \frac{d^2x}{dt^2}, \ldots, \frac{d^nx}{dt^n}\right) = 0$$

常微分方程式の分類 (7)

上述の微分方程式は, $x_1 = x, x_2 = \frac{dx}{dt}, \ldots, x_n = \frac{d^{n-1}x}{dt^{n-1}}$ とおくことにより, 次のように書ける.

$$\frac{dx_1}{dt} = x_2$$

$$\ldots$$

$$\frac{dx_{n-1}}{dt} = x_n$$

$$h\left(t, x_1, \ldots, x_n, \frac{dx_n}{dt}\right) = 0$$

常微分方程式の分類 (8)

$$\boldsymbol{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix}, \quad \boldsymbol{g}\left(t, \boldsymbol{x}, \frac{d\boldsymbol{x}}{dt}\right) = \begin{pmatrix} \frac{dx_1}{dt} - x_2 \\ \vdots \\ \frac{dx_{n-1}}{dt} - x_n \\ h(t, x_1, \ldots, x_n, \frac{dx_n}{dt}) \end{pmatrix}$$

とおけば, 先の微分方程式は $\boldsymbol{g}\left(t, \boldsymbol{x}, \frac{d\boldsymbol{x}}{dt}\right) = \boldsymbol{0}$ となる.

常微分方程式の分類 (9)

- 以上のようにして, 未知関数の n 階までの微分を含む微分方程式は, 未知関数の 1 階までの微分を含む常微分方程式 (ただし変数はベクトルで式が複数) に書き直される.

常微分方程式の分類 (10)

- $h\left(t, \boldsymbol{x}, \frac{d\boldsymbol{x}}{dt}, \ldots, \frac{d^n\boldsymbol{x}}{dt^n}\right) = 0, h\left(t, x, \frac{dx}{dt}, \ldots, \frac{d^n x}{dt^n}\right) = 0,\ h\left(t, \boldsymbol{x}, \frac{d\boldsymbol{x}}{dt}, \ldots, \frac{d^n\boldsymbol{x}}{dt^n}\right) = 0$ についても同様に処理できるから …
- はじめから $\boldsymbol{g}(t, \boldsymbol{x}, \frac{d\boldsymbol{x}}{dt}) = \boldsymbol{0}$ という形の微分方程式を考えればよい.

常微分方程式の分類 (11)

- 以下しばらく, $\boldsymbol{x}$ がベクトルで式が複数あることを前提とし, 微分方程式などを $\boldsymbol{g}\left(t, \boldsymbol{x}, \frac{d\boldsymbol{x}}{dt}\right) = \boldsymbol{0}$ というふうに書く.
- $\boldsymbol{x}$ がスカラーである場合や式が 1 個の場合には, $\boldsymbol{x}$ を x で, $\boldsymbol{g}$ などを g などで読み換えて考えること.

常微分方程式の分類 (12)

- 微分方程式 $\boldsymbol{g}\left(t, \boldsymbol{x}, \frac{d\boldsymbol{x}}{dt}\right) = \boldsymbol{0}$ が $\frac{d\boldsymbol{x}}{dt}$ の項について解けるとき, すなわち, 適切な関数 $\boldsymbol{f}$ を定めることで, $\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}\left(t, \boldsymbol{x}\right)$ と変形できるとき, この微分方程式は **explicit(陽的)** であるという.
- explicit でない微分方程式のことを **implicit (陰的)** であるという.

常微分方程式の分類 (13)

- 微分方程式が (微分を含まない) 非線形方程式と連立されているとき, すなわち

$$\boldsymbol{g}\left(t, \boldsymbol{x}, \frac{d\boldsymbol{x}}{dt}\right) = \boldsymbol{0},$$

$$\boldsymbol{h}\left(t, \boldsymbol{x}\right) = \boldsymbol{0}$$

となっているとき, これを 微分代数方程式 (系) という.

常微分方程式の分類 (14)

- 微分代数方程式 (系) をディスクリプタシステムと呼ぶこともある.
- implicit な微分方程式は, 必ず explicit な形に変形できるとは限らない. そして, 部分的に explicit な形に書き直すと, 微分代数方程式があらわれることがある.

常微分方程式の分類 (15)

- 微分方程式に, さらに一定時間前の未知関数値の影響の項が含まれることがある. $\delta_1, \ldots, \delta_N$ を正のスカラーとしたとき,

$$\boldsymbol{g}\left(t, \boldsymbol{x}(t), \boldsymbol{x}(t-\delta_1), \ldots, \boldsymbol{x}(t-\delta_N), \frac{d\boldsymbol{x}}{dt}\right) = \boldsymbol{0}$$

となっている場合である. このようなシステムをハイブリッドシステムという.

常微分方程式の分類 (16)

- 後述するように, explicit な常微分方程式では, (一定の条件のもとで) 解の存在と一意性が保証される.
- これに対し, 微分代数系やハイブリッドシステムでは, 解が存在しないこともあり得る.

常微分方程式の分類 (17)

- パラメータ $\boldsymbol{p}$ が含まれる場合の式としては, 以下のようなものを考えればよい.

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}\left(t, \boldsymbol{x}, \boldsymbol{p}\right),$$
$$\boldsymbol{g}\left(t, \boldsymbol{x}, \frac{d\boldsymbol{x}}{dt}, \boldsymbol{p}\right) = \boldsymbol{0},$$
$$\boldsymbol{h}\left(t, \boldsymbol{x}, \boldsymbol{p}\right) = 0,$$
$$\boldsymbol{g}\left(t, \boldsymbol{x}(t), \boldsymbol{x}(t-\delta_1), \ldots, \boldsymbol{x}(t-\delta_N), \frac{d\boldsymbol{x}}{dt}, \boldsymbol{p}\right) = \boldsymbol{0}.$$

常微分方程式の分類 (18)

- この講義では微分代数系やハイブリッドシステムは取り扱わない．それにもかかわらずこれらの名前を挙げたのは, Scilab における微分方程式関連の関数に, 微分代数系用のものと, ハイブリッドシステム用のものが含まれるからである．

常微分方程式の分類 (19)

- 以下の議論では, 高階の微分方程式を 1 階の微分方程式に直す操作はすでに終わっているものとし, $\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(\boldsymbol{x}, t)$ あるいは $\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(\boldsymbol{x}, t, \boldsymbol{p})$ のみを検討の対象とする.

- $\boldsymbol{x}$ および $\boldsymbol{f}$ はスカラーであってもよいが, $\boldsymbol{x}$ と $\boldsymbol{f}$ の次元は一致していなければならない.

解の存在と一意性 (1)

- 微分方程式 $\dfrac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(\boldsymbol{x}, t)$ の解を区間 $t \in [a, b]$ で求める問題を考える.
- a を初期時刻, b を終了時刻という.
- 微分方程式 $\dfrac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(\boldsymbol{x}, t)$ を満たす未知関数の中に, $t = a$ において値が $\boldsymbol{x}_a$ となるものがあれば, これを $\boldsymbol{\varphi}(t, a, \boldsymbol{x}_a)$ と書く.

解の存在と一意性 (2)

- $\boldsymbol{\varphi}(t, a, \boldsymbol{x}_a)$ の存在性については後述.
- $\boldsymbol{\varphi}(t, a, \boldsymbol{x}_a)$ が存在したとしても, ある t に対して $\boldsymbol{\varphi}(t, a, \boldsymbol{x}_a)$ が一意的に定まらない場合には, これは関数とはいえない. $\boldsymbol{\varphi}(t, a, \boldsymbol{x}_a)$ が関数であることについても後述.
- 上記の $\boldsymbol{x}_a$ を微分方程式の初期値という.

解の存在と一意性 (3)

- 初期値 $\boldsymbol{x}_a$ に対応する $\boldsymbol{\varphi}(t, a, \boldsymbol{x}_a)$ を求める問題を, 微分方程式の初期値問題という.
- 初期値 $\boldsymbol{x}_0$ の成分の一部と $t = b$ において解が取るべき値の成分の一部が指定されていて, $t = a$ と $t = b$ において該当する成分が指定された値を取る $\boldsymbol{\varphi}(t, a, \boldsymbol{x}_0)$ を求める問題を, 微分方程式の境界値問題という.

解の存在と一意性 (4)

- これから述べるように, 一定の条件の下で初期値問題は解を持つが, 境界値問題の解は必ずしも存在するとは限らない.
- 境界値問題を考えるのは, いくつかの重要な応用問題 (たとえば最適制御) において境界値問題があらわれるからであるが, この講義では常微分方程式の境界値問題は取り扱わない.

解の存在と一意性 (5)

$\boldsymbol{x} \in \mathbb{R}^n$, $D = \{(t, \boldsymbol{x}) : |t - t_0| \leq a, \|\boldsymbol{x} - \boldsymbol{x}_0\| \leq b\}$, G は D を含む開集合, $\boldsymbol{f} : G \to \mathbb{R}^n$ とする. $\boldsymbol{f}$ は D 上連続で, t に関して一様に $\boldsymbol{x}$ について Lipschitz 連続 ($\exists K > 0$, $\forall (t, \boldsymbol{x}_1), (t, \boldsymbol{x}_2) \in D$, $\|\boldsymbol{f}(t, \boldsymbol{x}_2) - f(t, \boldsymbol{x}_1)\| \leq K\|\boldsymbol{x}_2 - \boldsymbol{x}_1\|$) とし, $M = \max_{(t,x)\in D} \|f(t, x)\|$, $h = \min\{a, b/M\}$ とする. このとき, 微分方程式 $\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(t, \boldsymbol{x})$, $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$ は区間 $\{t : |t - t_0| \leq h\}$ において解を持つ.

解の存在と一意性 (6)

先に述べた微分方程式の解は初期値に対して一意的に定まる. また, その解は初期値に関して連続である.

パラメータを含む微分方程式 $\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{p})$, $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$ については, $\boldsymbol{f}$ が D 上連続で, t および $\boldsymbol{p}$ に関して一様に $\boldsymbol{x}$ について Lipschitz 連続であると仮定すれば, 解の存在と一意性, 初期値およびパラメータに関する連続性が導かれる.

さらに, 一定の条件のもとで, 解の初期値やパラメータに関して微分可能であることを示すことができる.

解の存在と一意性 (7)

- 以上により, 上述の仮定のもとで, 常微分方程式の初期値問題は必ず解 $\boldsymbol{\varphi}(t, a, \boldsymbol{x})$ を持ち, これが t の (微分可能) 関数として定まることがわかる.
- この講義では微分方程式が一意解を持ち, 必要な回数だけ微分可能であると仮定する.
- 先に述べたように, 高階の微分方程式は 1 階の (連立) 微分方程式 (系) に書き直せるので, 高階の微分方程式について改めて議論する必要はない.

Scilab の常微分方程式関連関数一覧 (1)

`ode`	1 階の常微分方程式の初期値問題を解く. オプションによって Adams 法, BDF 法, 適応 Runge-Kutta 法, Fehlberg 法, 一般的な差分方程式, 終端条件付きの求解を選択できる.
`bvode`	常微分方程式の境界値問題を解く (1 階とは限らない).

この講義では常微分方程式の境界値問題は取り扱わない.

Scilab の常微分方程式関連関数一覧 (2)

`dae`	微分代数系のソルバ
`daskr`	同上
`dasrt`	同上
`odedc`	ハイブリッドシステムのソルバ

この講義では微分代数系やハイブリッドシステムは取り扱わない.

1 段法 (1)

- この講義では, 未知関数 $\boldsymbol{x}(t) = \boldsymbol{\varphi}(t, t_0, \boldsymbol{x}_0)$ がベクトル値関数の場合をおもに取り扱う.
- 常微分方程式を数値解法に解くときには, $[a, b]$ を適当な分点 $a = t_0 < t_1 < \cdots < t_N = b$ で分割し, 各 t_k において未知関数の値 $\boldsymbol{x}(t_k)$ を求めるという方法が取られることが多い. この方法を離散変数法という (岩波数学辞典).

1 段法 (2)

- $h_k = t_{k+1} - t_k$ と定義し, これをステップ幅あるいは刻み幅呼ぶ ([山本]). $t_k + h_k = t_{k+1}$ となることに注意. ステップ幅が k によらず一定のときには, これを h と書く.
- 以下では, $\boldsymbol{x}_k = \boldsymbol{x}(t_k)$ と書く $(0 \leq k \leq N)$.
- 各 k における $\boldsymbol{x}_k$ の近似を $\boldsymbol{\xi}_k$ とする.

1 段法 (3)

- 離散変数法は, 未知の列 $(\boldsymbol{x}_k)_{k=1,2,\ldots,N}$ を近似する別の列 $(\boldsymbol{\xi}_k)_{k=1,2,\ldots,N}$ を構成する方法の総称である.
- 近似列 $(\boldsymbol{\xi}_k)_{k=1,2,\ldots,N}$ は, 多くの場合, 差分方程式を解くことによって構成される.
- $\boldsymbol{x}_k$, $\boldsymbol{\varphi}$, $\boldsymbol{f}$ の第 j 成分を $x_{k,j}$, φ_j, f_j と書く.

1 段法 (4)

- 中間値の定理より, ある $c \in [t_k, t_{k+1}]$ に対し, $x_{k+1,j} - x_{k,j} = \frac{d}{dt}\varphi\big|_{t=c} h_k = f_j(\boldsymbol{\varphi}(c, t_k, \boldsymbol{x}_k), c)h_j$ である. c の値は $\boldsymbol{x}_k$, $\boldsymbol{x}_{k+1}$, t_k, t_{k+1} で決まる.
- $\phi_j(t_k, t_{k+1}, \boldsymbol{x}_k, \boldsymbol{x}_{k+1}; h_k) = f_j(\boldsymbol{\varphi}(c, t_k, \boldsymbol{x}_k), c)$ と定義し, ϕ_j を成分として持つベクトル値関数を $\boldsymbol{\Phi}$ とすると,
  $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + h_k \boldsymbol{\Phi}(t_k, t_{k+1}, \boldsymbol{x}_k, \boldsymbol{x}_{k+1}; h_k)$.

1 段法 (5)

- $\boldsymbol{\Phi}$(未知) を近似する関数を $\boldsymbol{\Psi}$ とすると,
  $\boldsymbol{x}_{k+1} \simeq \boldsymbol{x}_k + h_k \boldsymbol{\Psi}(t_k, t_{k+1}, \boldsymbol{x}_k, \boldsymbol{x}_{k+1}; h_k)$.

- 上記の $\boldsymbol{x}_{k+1}$, $\boldsymbol{x}_{k+1}$ を $\boldsymbol{\xi}_{k+1}$, $\boldsymbol{\xi}_{k+1}$ で置き換え, $\simeq$ を等号に変えたものが, **1 段法**と呼ばれる公式の一般形 (次式) である.

$$\boldsymbol{\xi}_{k+1} = \boldsymbol{\xi}_k + h_k \boldsymbol{\Psi}(t_k, t_{k+1}, \boldsymbol{\xi}_k, \boldsymbol{\xi}_{k+1}; h_k)$$

1 段法 (6)

- 先に述べた公式は両辺に $\boldsymbol{\xi}_{k+1}$ を含んでいる. このように, 両辺に同じ変数を含んだ公式を, **implicit(陰的)** な公式という.
- この公式で数値解を構成しようとすると, 各ステップで非線形方程式を反復法によって解く必要が生じることがある. これを内部反復という.

1 段法 (7)

- implicit な公式を使った方がうまく解ける問題が存在するため (**硬い (stiff)** という ([Hairer et al.])), この公式には存在意義があるが (硬い系の定義は文献によって異なる), 内部反復が必要であるため, 必ずしも使いやすいとはいえない.

1 段法 (8)

- 近似関数 $\boldsymbol{\Psi}$ を構成する際に変数 t_{k+} と $\boldsymbol{\xi}_{k+1}$ を使うのを止めれば, 内部反復の必要はなくなる. このようにした公式を **explicit(陽的)** な公式という. 一般形を次に示す.

$$\boldsymbol{\xi}_{k+1} = \boldsymbol{\xi}_k + h_k \boldsymbol{\Psi}(t_k, \boldsymbol{\xi}_k, ; h_k)$$

1 段法 (9)

- 近似関数 Ψ の構成法は様々であるが, 今回の講義では単純な形のもののみを紹介し, より精密な方法については次回に述べる.

1 段法 (10)

- $\boldsymbol{\xi}_{k+1}$ を構成するために, $\boldsymbol{\xi}_k$ だけでなく, 過去の数値解の系列 $(\boldsymbol{\xi}_{k-L}, \ldots, \boldsymbol{\xi}_k)$ および対応する関数値 $\left(\boldsymbol{f}(\boldsymbol{\xi}_{k-L}, t_{k-L}), \ldots, \boldsymbol{f}(\boldsymbol{\xi}_k, t_k)\right)$ を使う方法もある. このような方法を多段法という.
- 多段法については次回述べる.

Euler 法 (1)

- 常微分方程式の数値解法のうち, もっとも基礎的なものが Euler 法である.
- **(前進)Euler 法**は, $\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(\boldsymbol{x}, t)$ の解を, 差分方程式 $\boldsymbol{\xi}_{k+1} = \boldsymbol{\xi}_k + h_k \boldsymbol{f}(\boldsymbol{\xi}_k, t_k)$ を解くことで構成する方法である. これは 1 段法で, explicit である.

Euler 法 (2)

- 前進 Euler 法は, 先の中間値の定理を使った議論で $c = t_k$ として近似関数を構成した場合に相当する.
- h_k をどんどん細かくすると, Euler 法が生成する近似解は微分方程式 $\dfrac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(\boldsymbol{x}, t)$ の解に収束することが示せる ([Hairer et al.]).

Euler 法 (3)

- 1 段法の特徴は h が大きくなるにしたがって数値的な近似解と微分方程式の真の解のあいだの誤差が増大することであり, Euler 法ではこの問題が顕著である. この意味で, 高精度の近似解が必要な場合には, Euler 法は必ずしも適しているとはいえない.

Euler 法 (4)

- Euler 法を改良したものに, **Heun 法**と呼ばれる方法がある. これは,

$$\boldsymbol{z}_{k+1} = \boldsymbol{\xi}_k + h_k f(\boldsymbol{\xi}_k, t_k),$$
$$\boldsymbol{\xi}_{k+1} = \boldsymbol{\xi}_k + \frac{h}{2}\left(\boldsymbol{f}(\boldsymbol{z}_{k+1}, t_{k+1}) + \boldsymbol{f}(\boldsymbol{\xi}_k, t_k)\right)$$

とする方法である (文献によって呼び方が変わることがある).

Euler 法 (5)

- **後退 Euler 法**は, $\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(\boldsymbol{x}, t)$ の解を, 差分方程式 $\boldsymbol{\xi}_{k+1} = \boldsymbol{\xi}_k + h_k \boldsymbol{f}(\boldsymbol{\xi}_{k+1}, t_{k+1})$ により構成する方法である. implicit であること以外は前進 Euler 法と同じである. 中間値の定理を使った議論で $c = t_{k+1}$ とした場合に相当する.
- Scilab でこれらの挙動を確認する.

Scilab における Euler 法 (1)

- 関数 `ode` のオプションは Euler 法を含まないので, Euler 法を実行するには, 冒頭で`"discrete"`と宣言して, 自分で対応する関数を定義する. この方法は, 1 段法であれば何でも (自分で適合する関数を定義すれば) 実行できる方法である.
- `"discrete"`を宣言せずに `ode` を使う方法はこれとは異なるので, 混乱しないよう注意すること.

Scilab における Euler 法 (2)

- 以下の線形微分方程式を解く問題を考える.
  $\frac{d\boldsymbol{x}}{dt} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \boldsymbol{x}, \quad \boldsymbol{x}(0) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$
  解は $x_1(t) = \cos t$, $x_2(t) = \sin t$ である. これと前進 Euler 法, Heunr 法, 後退 Euler 法による近似解 (ステップ幅 h) を比較する.
- $\boldsymbol{A} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$, $\boldsymbol{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ とおく.

Scilab における Euler 法 (3)

各アルゴリズムは以下のようになる.

前進 Euler 法	$\boldsymbol{\xi}_{k+1} = (\boldsymbol{I} + h\boldsymbol{A})\boldsymbol{\xi}_k$
Heun 法	$\boldsymbol{z}_{k+1} = (\boldsymbol{I} + h\boldsymbol{A})\boldsymbol{\xi}_k,$ $\boldsymbol{\xi}_{k+1} = \boldsymbol{\xi}_k + \frac{h}{2}\left(\boldsymbol{A}(\boldsymbol{\xi}_k + \boldsymbol{z}_{k+1})\right)$
後退 Euler 法	$\boldsymbol{\xi}_{k+1} = \frac{1}{1+h^2}\left(\boldsymbol{I} + h\boldsymbol{A}\right)\boldsymbol{\xi}_k$

次ページにプログラムを示す ($h = 0.1$, $t \in [0, 20]$).

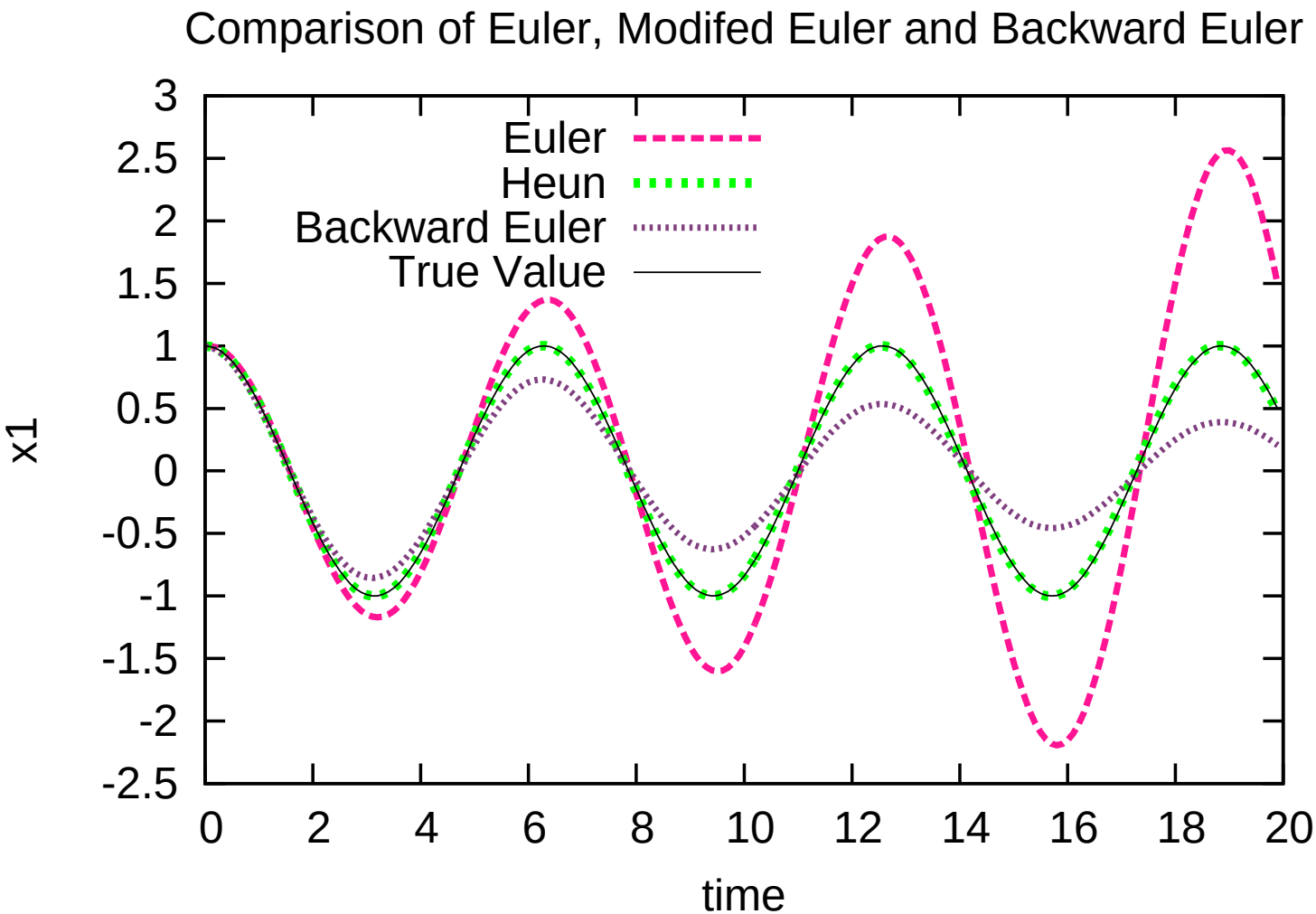
```
//前進 Euler 法
h=0.1;
deff('y=f(t,x)','y=[1 -h;h 1]*x');
x0=[1;0];
tLen=200;
x=ode("discrete",x0,1,1:tLen,f);
```

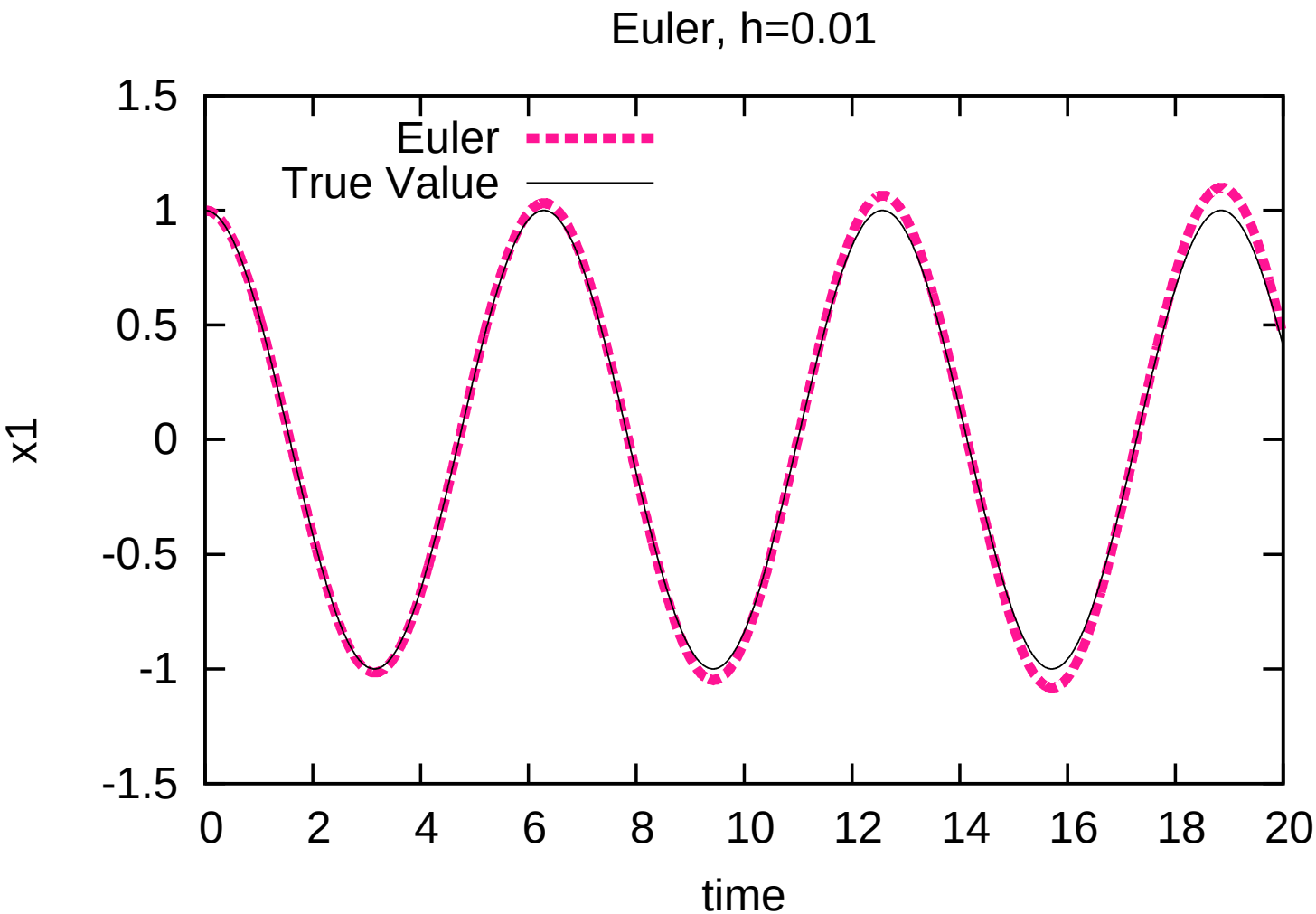
```
//後退 Euler 法
h=0.1;
deff('y=f(t,x)','y=([1 -h;h 1]*x)/(1+h^2)');
x0=[1;0];
tLen=200;
x=ode("discrete",x0,1,1:tLen,f);
```

```
//Heun 法
h=0.1;
deff('y=g(x)','y=[0 -1;1 0]*x');
function y=f(t,x)
  w=g(x);
  z=x+h*w;
  y=x+h*(w+g(z))/2;
endfunction

x0=[1;0];
tLen=200;
x=ode("discrete",x0,1,1:tLen,f);
```

次ページに $x_1(t)$ の数値解と真値 $(\cos t)$ との比較を示す. Heun 法と比較して, 前進 Euler 法および後退 Euler 法の精度は悪いことがわかる.


Comparison of Euler, Modifed Euler and Backward Euler
Euler
Heun
Backward Euler
True Value
x1
time
3
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
0
2
4
6
8
10
12
14
16
18
20


Euler, h=0.01
Euler
True Value
x1
time
1.5
1
0.5
0
-0.5
-1
-1.5
0
2
4
6
8
10
12
14
16
18
20

Scilab の ode の使い方に関する注意 (1)

- キーワード"discrete"を指定した場合には, f の部分には数値解法の公式を指定する.
- キーワード"discrete"を指定しない場合 (Scilab に用意されている公式を使う場合) には, f の部分には微分方程式の右辺の関数を指定する.

Scilab の ode の使い方に関する注意 (1)

次回の先取りになるが, 以下と"discrete"ありの場合を比較すること.

```
function dx=f(t,x)
  dx=[0 -1;1 0]*x;
endfunction

t0=0;
t=0:.1:100;
x0=[1;0];
y=ode(x0,t0,t,f); //デフォルト
y=ode("adams",x0,t0,t,f); //Adams 法
y=ode("stiff",x0,t0,t,f); //BDF 法
y=ode("rk",x0,t0,t,f);    //Runge-Kutta 法
y=ode("rkf",x0,t0,t,f);   //RKF45
```