数理生物学演習

第４回　個体群動態の数理モデル（２）：
指数増殖モデル，ロジスティック成長モデル

野下 浩司（Noshita, Koji）

noshita@morphometrics.jp

https://koji.noshita.net

理学研究院 数理生物学研究室

第４回：指数増殖，ロジスティック成長

本日の目標

- 微分方程式を数値的に解く
- 結果を解析解と比較する

指数増殖

ある集団のサイズ（個体数）を x とし，
その増加速度（dx/dt）が集団サイズ $x(t)$ に比例する場合，
ダイナミクスは以下の式で表すことができる．

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

初期条件

$$x(0) = x_0$$

a：単位時間あたり一個体あたりの増加率（マルサス係数）

$$x(t) = x_0 e^{at}$$

解いてみよう

ロジスティック成長

指数増殖におけるマルサス係数（a）がr(1-x/K)で置き換えられている.
つまり，個体数が増加するに伴い単位時間あたり一個体あたりの増加率が減少する.

$$\frac{dx}{dt} = r\left(1 - \frac{x}{K}\right)x$$

初期条件

$$x(0) = x_0$$

r：内的自然増加率．個体群密度が十分に小さい（~0）場合の増加率.
K：環境収容力．ある環境が維持できる個体数．利用できる資源量や空間サイズなどを反映する．ここでは$K > 0$とする.

$$x(t) = \frac{K}{1 + \left(\frac{K}{x_0} - 1\right)e^{-rt}}$$

解いてみよう

条件分岐

条件分岐 if文

特定の条件下でのみ実行したい処理を書く！

・if文

```
if 条件文:
    文１
    文2
    ⋮
    文n
```

条件文が真ならば，
文１～文nを実行する．
（偽ならば何もしない）

特に注意！！
インデントされた文が
if文のブロックとみなされる

```
# 01-01. if文

for i in range(30):
    if i > 15:
        print(i)
```

出力
16
17
⋮
29

iが15より大きければ
iを画面に表示する

forループとif文が使えればたいていのプログラムが組める！

関係演算子と論理演算子

```
# 01-02. 関係演算子と論理演算子

for i in range(10):
    print("i=", i)
    if i > 5:
        print("  iは5より大きい")

    if i == 3:
        print("  iは3と等しい")

    if i>=3 and i<=6:
        print("  iは3以上6以下")

    if not (i==1 or i==2):
        print("  iは1または2ではない")
```

出力

```
i= 0
  iは1または2ではない
i= 1
i= 2
i= 3
  iは3と等しい
  iは3以上6以下
  iは1または2ではない
i= 4
  iは3以上6以下
  iは1または2ではない
i= 5
  iは3以上6以下
  iは1または2ではない
i= 6
  iは5より大きい
  iは3以上6以下
  iは1または2ではない
i= 7
  iは5より大きい
  iは1または2ではない
i= 8
  iは5より大きい
  iは1または2ではない
i= 9
  iは5より大きい
  iは1または2ではない
```

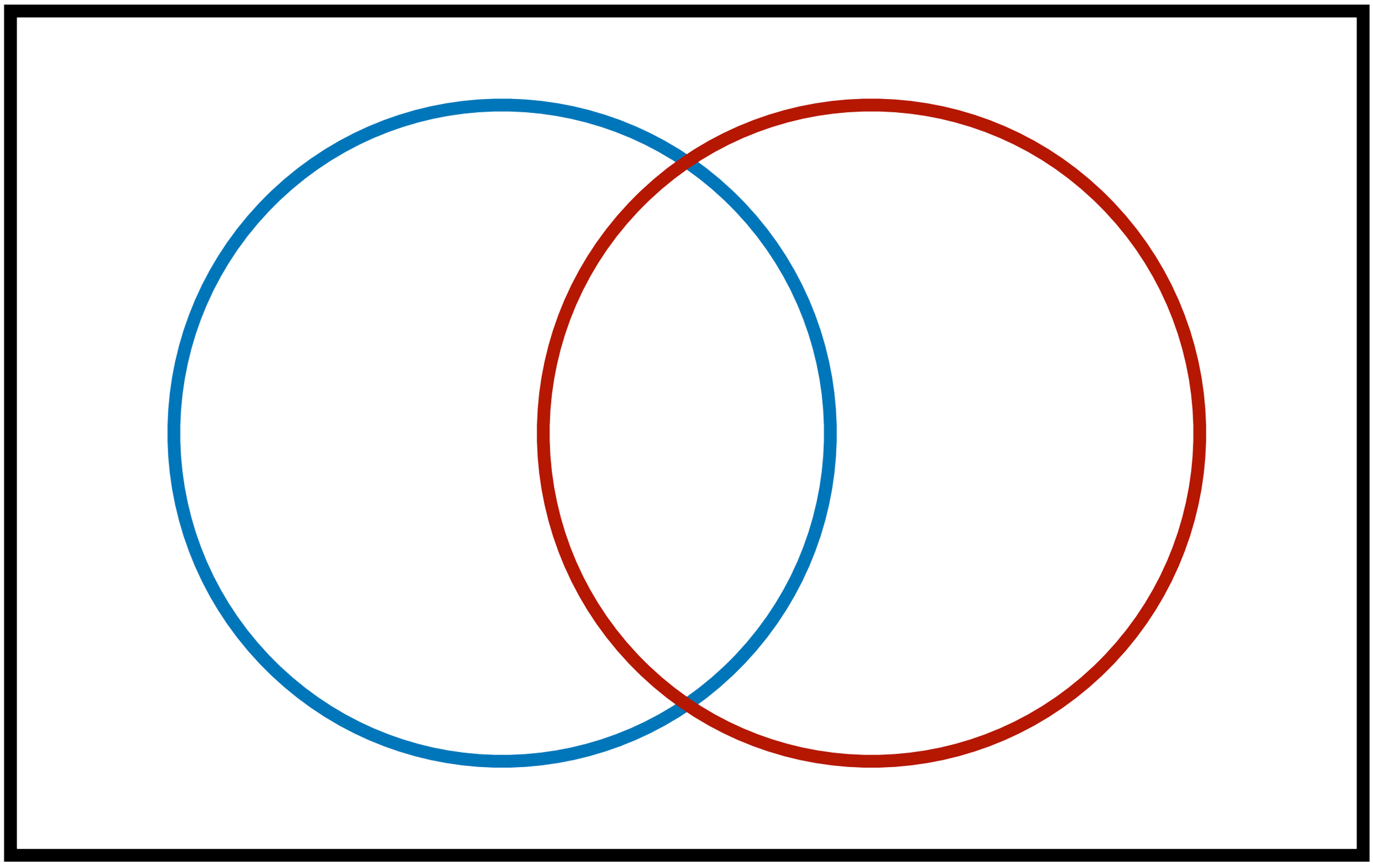
関係演算子

- \>　より大きい
- >=　以上
- <　より小さい
- <=　以下
- ==　等しい
- !=　等しくない

論理演算子

- and　かつ（論理積）
- or　または（論理和）
- not　ではない（否定）

関係演算子と論理演算子



悩んだらベン図を描いてみる

if文を鍛える（1）

・if-else文

```
if 条件文:
  文1
  文2
  ⋮
  文m
else:
  文'1
  文'2
  ⋮
  文'n
```

条件文が
真ならば文１～文mを実行
偽ならば文'１～文'nを実行

```
# 01-03. 偶奇判定
for i in range(50):
    if i%2==0:
        print("even")
    else:
        print("odd")
```

iが偶数ならばevenを
奇数ならばoddを
画面に出力する

出力

```
even
odd
even
odd
・・・
odd
```

!注意：elseはifと同じ深さ

if文を鍛える（2）

・if-else-if文

```
if条件文1:
  文１
  文2
  ⋮
  文m
elif 条件文2:
  文’1
  文’2
  ⋮
  文’n
```

条件文1が真ならば文１～文mを実行
条件文1が偽かつ条件文2が真ならば文’1～文’nを実行
（条件文1も2も偽ならば何もしない）

```
# 01-04. 3の倍数
for i in range(50):
    if i%3==0:
        print("3の倍数")
    elif i%3==1:
        print("余り1")
    else:
        print("")
```

出力

```
余り1

3の倍数
余り1

3の倍数
・・・
余り1
```

iを3で割った余りが
0のとき，「３の倍数」
1のとき，「余り1」
をそれぞれ画面に表示し，
2のとき，何もしない

分岐図 ifバージョン

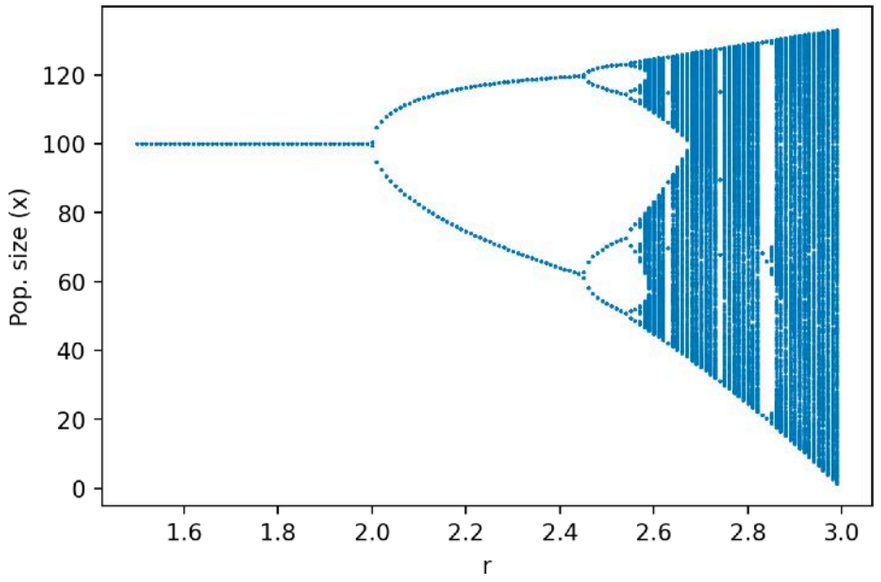
```
# 01-05. 分岐図 ifバージョン
import matplotlib.pyplot as plt

K = 100

r_list = []
x_list = []

for i in range(150,300):
    x0 = 10
    r = i/100
    x=x0
    for t in range(5000):
        xx =x + r* (1-x/K)*x
        x = xx
        if t >= 4000:
            r_list.append(r)
            x_list.append(x)

plt.figure(dpi = 200)
plt.plot(r_list, x_list, '.', markersize="1")
```



rを1.5から3まで0.01刻みで変化させる

今回は全部で5000ステップ計算する

最後の1000ステップをリストに格納

様々なrの値ループ

（特定のrについての）ダイナミクスの計算ループ

解析解のプロット

指数増殖（解析解）

```python
# 02-01. 指数増殖

import math
import matplotlib.pyplot as plt

a = 0.2
x0 = 10

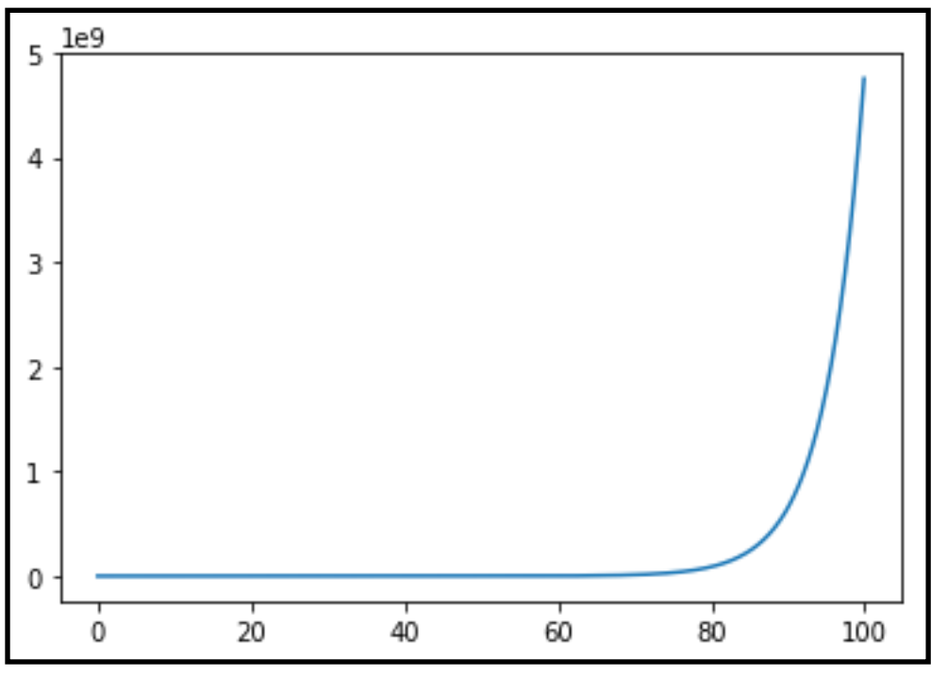
dt = 0.1
t_list = []
x_list = []
for i in range(1000):
    t = dt*i
    x = x0*math.exp(a*t)
    t_list.append(t)
    x_list.append(x)

plt.plot(t_list, x_list)
```

モジュールやパッケージは
ノートブック中で１度読み込めばOK

いろいろなオプションを調整して
見やすくプロットしてみよう.

解析解
$x(t) = x_0 e^{at}$

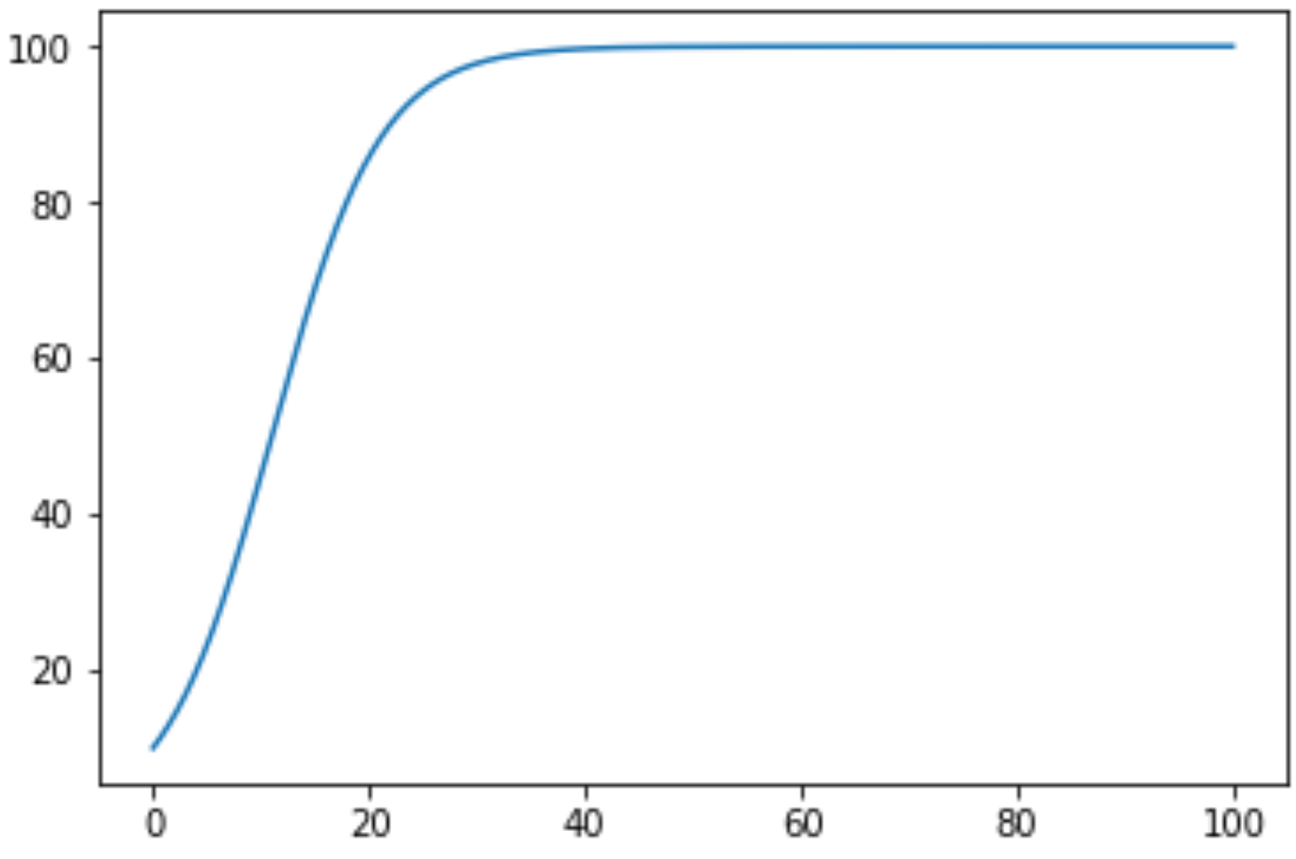


！注意：dtを変化させたら，その分iの最大値を変えなければ，観察している時間の長さが変わる.

ロジスティック成長（解析解）

```
# 02-02. ロジスティック成長
```

指数増殖のプログラムを参考に自分で考えてみてください



こんな感じのがプロットしたい

微分方程式を数値的に解く

オイラー法 Euler's method

- 計算機は直接は微分や積分ができない
- 微分方程式を（時間方向に）離散化し計算機が扱えるようにする

目的

微分方程式

$$\frac{dx}{dt} = f(x,t) \quad \cdots \quad (1)$$

を数値的に解きたい.

微分の定義から

$$\frac{dx}{dt} = \lim_{\Delta t \to 0} \frac{x(t+\Delta t) - x(t)}{\Delta t}$$

なので, Δt が十分に小さければ,

(1) は近似的に

$$f(x,t) \approx \frac{x(t+\Delta t) - x(t)}{\Delta t}$$

変形すると

$$x(t+\Delta t) \approx x(t) + f(x,t)\Delta t$$

$x(0) = x_0$とし,

$x_1\left(=x(\Delta t)\right), x_2\left(=x(2\Delta t)\right), \cdots, x_n\left(=x(n\Delta t)\right)$を考えると

$$x_n \approx x_{n-1} + f(x_{n-1}, t_{n-1})\Delta t$$

ただし, $t_n = \Delta t \cdot n$

ここで

$$X_n = X_{n-1} + f(X_{n-1}, t_{n-1})\Delta t$$

として, この X_n を x_n の近似値として採用する.

- 時間方向の刻み幅 Δt を小さくすることである程度誤差を小さくできる
- オイラー法はあまり精度の良い近似法ではない

最終的なプログラムは
前回の差分方程式と似たものになる

指数増殖の離散化

指数増殖

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

微分の近似（Δtは十分小さいとする）

$$\frac{dx}{dt} \approx \frac{x(t+\Delta t) - x(t)}{\Delta t}$$

↓

$$ax(t) \approx \frac{x(t+\Delta t) - x(t)}{\Delta t}$$

式を整理

$$x(t+\Delta t) \approx x(t) + ax(t)\Delta t$$

↓

$x(0) = x_0$ とし，　$x_1, x_2, \ldots, x_n$

また，　$t_n = \Delta t \cdot n$

$$x_{n+1} \approx x_n + ax_n\Delta t$$

↓

$$X_{n+1} = X_n + aX_n\Delta t$$

指数増殖

```
# 02-03. 指数増殖（数値解）

a = 0.2
x0 = 10

dt = 0.1

t = 0
x = x0
t_list = [t]
x_list = [x]
for i in range(1000):
    t = dt*(i+1)
    x = x + a*x*dt

    t_list.append(t)
    x_list.append(x)

plt.plot(t_list, x_list)
```

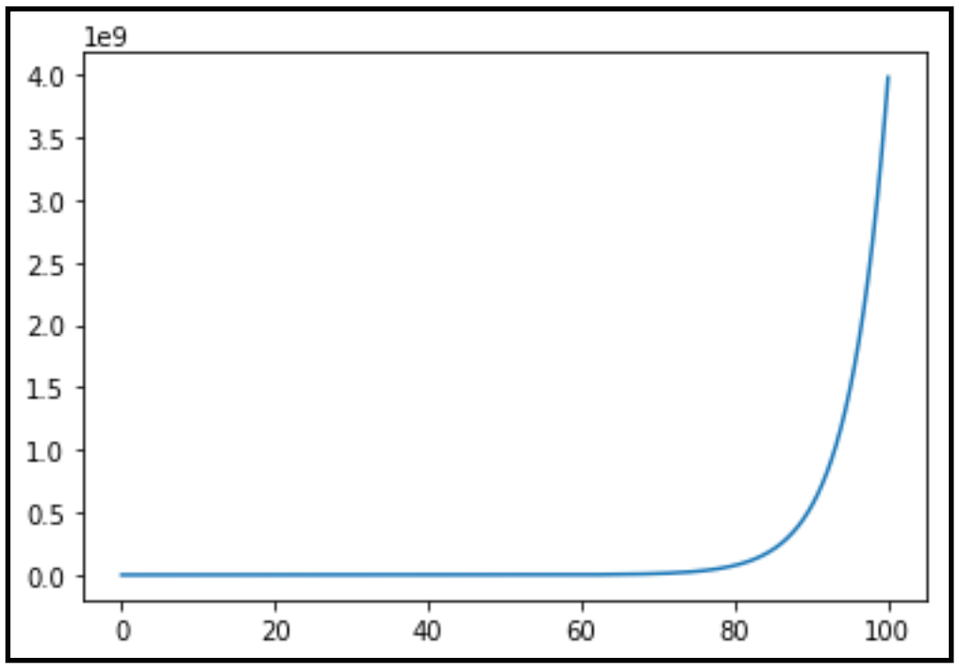
パラメータ・初期値の設定

時間方向の刻み幅の設定

オイラー法による近似

$$X_{n+1} = X_n + aX_n\Delta t$$

いろいろなオプションを調整して見やすくプロットしてみよう.



！注意：dtを変化させたら，その分iの最大値を変えなければ，観察している時間の長さが変わる.

指数増殖

```
# 02-04. 指数増殖（解析解と数値解の比較）

a = 0.2
x0 = 10

tEnd = 100

#
# 時間幅
#
dt_A = 0.1
iEndA = int(tEnd/dtA)+1

dt_N = 0.1
iEndN = int(tEnd/dtN)

# 解析解

x_list_A = []
t_list_A = []
for i in range(iEndA):
    t = dtA*i
    x = x0*math.exp(a*t)
    t_list_A.append(t)
    x_list_A.append(x)
```

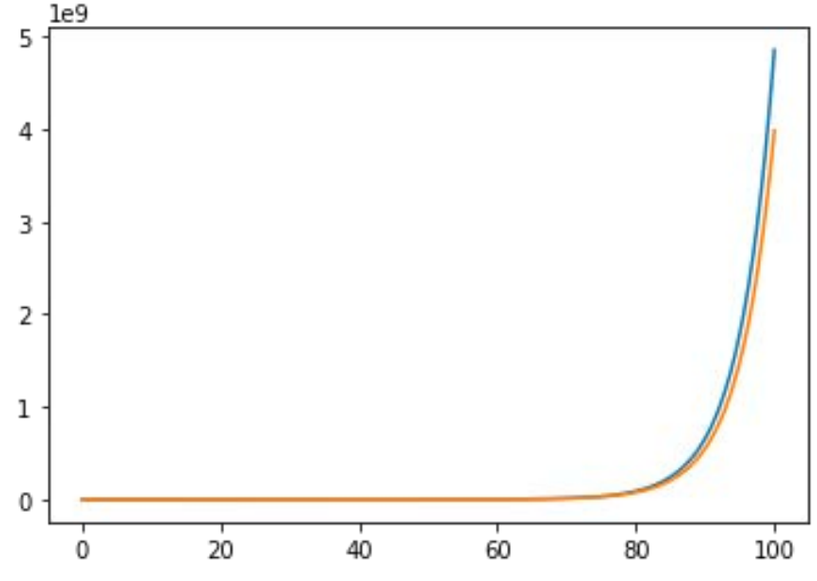
dt_Nを0.1～0.001の範囲で変えて，どの程度解析解と合うかを検討してみる

```
# 数値解

t = 0
x = x0
t_list_N = [t]
x_list_N = [x]
for i in range(iEndN):
    t = dtN*(i+1)
    xx = x + a*x*dtN
    x = xx
    t_list_N.append(t)
    x_list_N.append(x)

plt.plot(t_list_A, x_list_A,
         t_list_N, x_list_N)
```

出力例



ロジスティック成長の離散化

補足

ロジスティック成長

$$\frac{dx}{dt} = r\left(1 - \frac{x(t)}{K}\right)x(t)$$

微分の近似（Δtは十分小さいとする）

$$\frac{dx}{dt} \approx \frac{x(t+\Delta t) - x(t)}{\Delta t}$$

$$r\left(1 - \frac{x(t)}{K}\right)x(t) \approx \frac{x(t+\Delta t) - x(t)}{\Delta t}$$

式を整理

$$x(t+\Delta t) \approx x(t) + \Delta t \cdot r\left(1 - \frac{x(t)}{K}\right)x(t)$$

$x(0) = x_0$ とし，　$x_1, x_2, \ldots, x_n$

また，　$t_n = \Delta t$ ・ n

$$x_{n+1} \approx x_n + \Delta t \cdot r\left(1 - \frac{x_n}{K}\right)x_n$$

$$X_{n+1} = X_n + \Delta t \cdot r\left(1 - \frac{X_n}{K}\right)X_n$$

ロジスティック成長

```
#  ロジスティック成長　（数値解）
```

指数増殖のプログラムを参考に自分で考えてみてください

ロジスティック成長

課題ノーマル ２

#.ロジスティック成長　（解析解と数値解の比較）

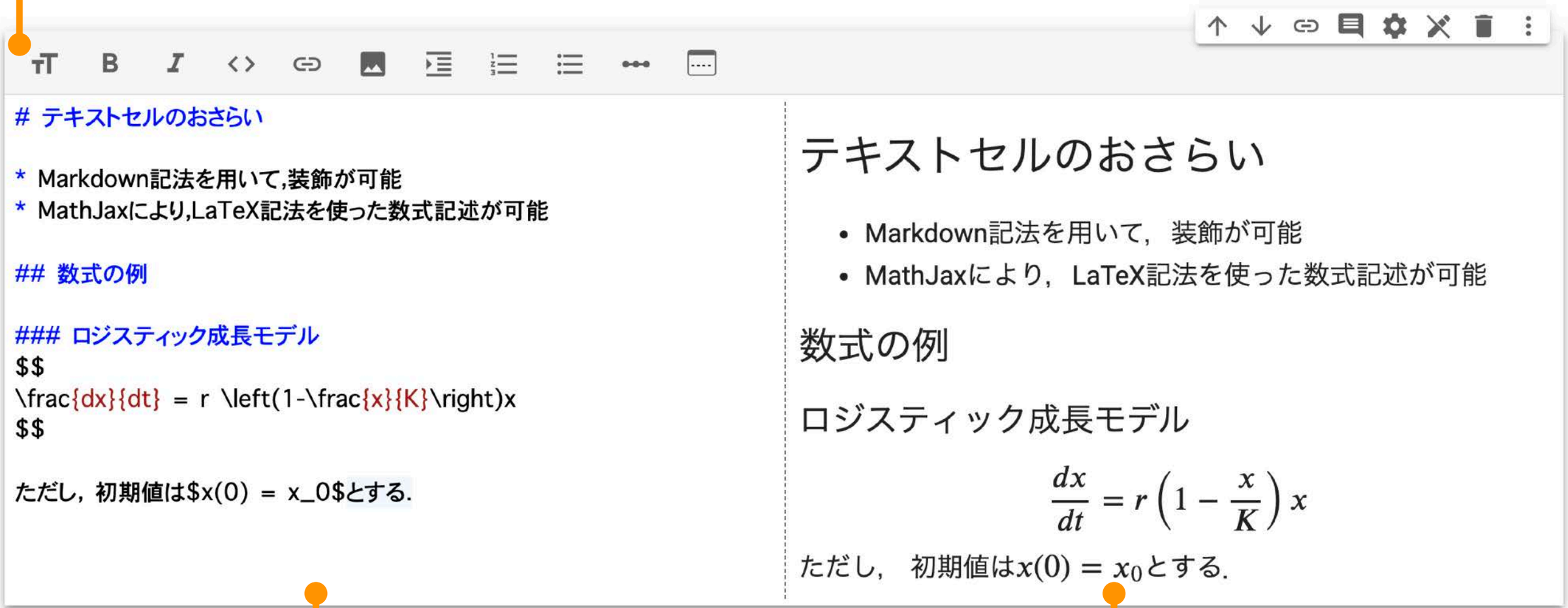
指数増殖のプログラムを参考に自分で考えてみてください

Colabのテキストセルに関する補足

Markdown，LaTeX記法

Colabのテキストセルのおさらい

各種オプション（レベル，太字，イタリック，コードブロック，リンク，…）



```
# テキストセルのおさらい

* Markdown記法を用いて,装飾が可能
* MathJaxにより,LaTeX記法を使った数式記述が可能

## 数式の例

### ロジスティック成長モデル
$$
\frac{dx}{dt} = r \left(1-\frac{x}{K}\right)x
$$

ただし，初期値は$x(0) = x_0$とする.
```

テキストセルのおさらい

- Markdown記法を用いて，装飾が可能
- MathJaxにより，LaTeX記法を使った数式記述が可能

数式の例

ロジスティック成長モデル

$$\frac{dx}{dt} = r\left(1 - \frac{x}{K}\right)x$$

ただし， 初期値は$x(0) = x_0$とする.

入力用フィールド

プレビュー用フィールド

- **Markdown記法を用いて，装飾が可能**
- **MathJaxにより，LaTeX記法を使った数式記述が可能**

Markdown

```
# 見出し

## 小見出し
`#`を重ねることで見出し，小見出しを作成できる．行頭で使う．

### （番号なし）リスト
`*`, `-`, `+`でリストを表現できる．

* 要素１
* 要素2
* 要素3

### 番号ありリスト
`数字.`で番号ありリストを作れる．番号は自動で振られる．

1. 要素１
1. 要素２
1. 要素３

### リンク
`[表示](URL)`でリンクを表現できる．

[演習用Webページ](https://koji.noshita.net/page/compbio/)

### コードとコードブロック
すでに利用しているが，
バッククォートで囲むとコードを表現する．
またバッククォートを3つ重ねて複数行を囲むとコードブロックが表現できる．
特定の言語のコードブロックの場合は最初の３連バッククォートのあとに言語名を記述する（対応している言語の場合はシンタックスハイライトが利用できる）．

`code`

```
import math
```

```python
import math
```
```

他にもいろいろな機能があるので興味ある人は調べて使おう．

- **参考**
  - GitHub Flavored Markdown Spec https://github.github.com/gfm/
  - Markdown書き方マニュアル（備忘録）https://sugarnaoming.github.io/markdown_manual/

Colabでの数式等の入力 LaTeX（1）

```
# $\LaTeX$を使った数式の表示（1）

## インライン数式とディスプレイ数式
インライン数式は文章中に数式を埋め込む．例えば，$f(x) = a x$な感じ．
ディスプレイ数式は式を新しい行にして（デフォルトでは中央揃えで）表示する：
$$ f(x) = a x $$
という感じ．

## 基本的な演算子
和 $+$ (`+`) や差 $-$ (`-`) の記号はそのまま．
ドット積 $\cdot$ (`\cdot`)，クロス積 $\times$ (`\times`) はこのあたりを使うと良い．

## 分数
`\frac{分子}{分母}`

$$
x = \frac{分子}{分母}
$$

インラインだと，高さが調整される．こんな $x = \frac{分子}{分母}$ 感じ．

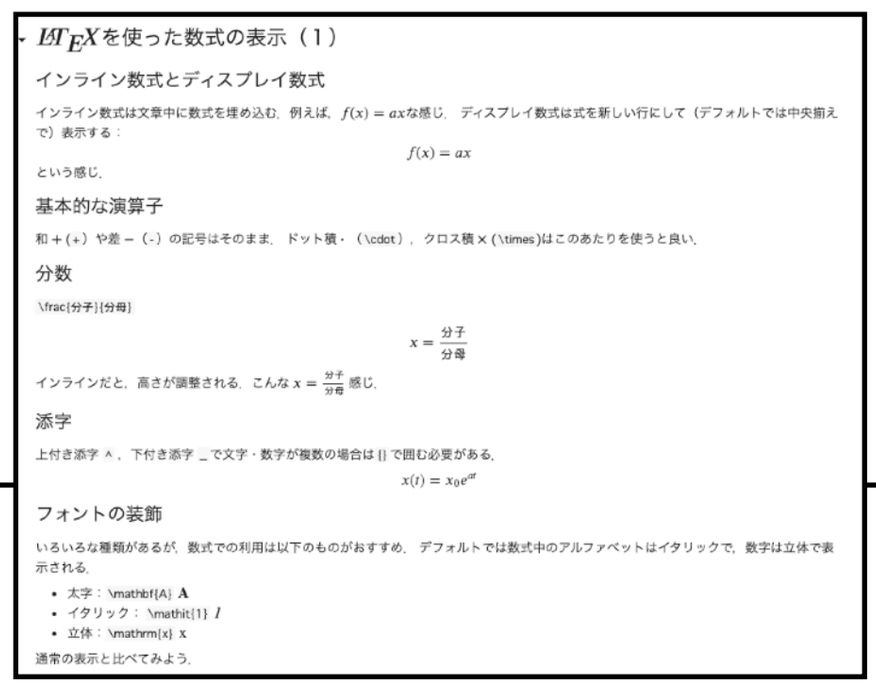
## 添字
上付き添字 `^` ，下付き添字 `_`で文字・数字が複数の場合は`{}`で囲む必要がある．

$$
x(t) = x_0 e^{a t}
$$

## フォントの装飾
いろいろな種類があるが，数式での利用は以下のものがおすすめ．
デフォルトでは数式中のアルファベットはイタリックで，数字は立体で表示される．

* 太字：`\mathbf{A}` $\mathbf{A}$
* イタリック： `\mathit{1}` $\mathit{1}$
* 立体：`\mathrm{x}` $\mathrm{x}$

通常の表示と比べてみよう．
```



- $数式$：インライン数式（文章の中に表示）
- $$数式$$：ディスプレイ数式（新しい行に表示）

Colabでの数式等の入力 LaTeX（2）

```
# $\LaTeX$を使った数式の表示（2）
## ギリシャ文字
`\名前`になっている．大文字は最初のアルファベットが大文字．

$\alpha, \beta, \gamma, \cdots$や$\Delta, \Phi$など．

## 改行
`\\`のようにバックスラッシュを２つ重ねる．
$$
f(x) = x^2 \\
g(x) = x^2 + 2 x + 1
$$

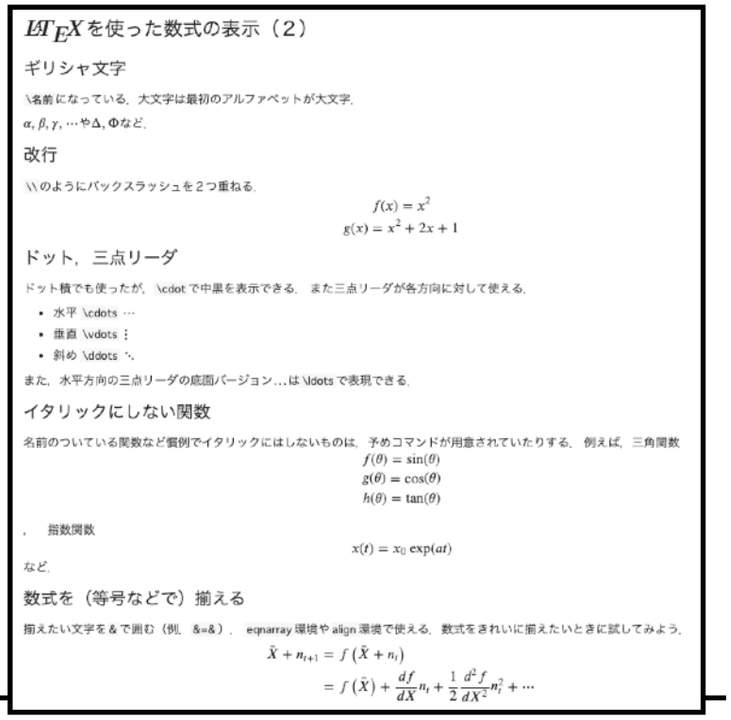
## ドット，三点リーダ
ドット積でも使ったが，`\cdot`で中黒を表示できる．
また三点リーダが各方向に対して使える．
* 水平 `\cdots` $\cdots$
* 垂直 `\vdots` $\vdots$
* 斜め `\ddots` $\ddots$

また，水平方向の三点リーダの底面バージョン$\ldots$は`\ldots`で表現できる．

## イタリックにしない関数
名前のついている関数など慣例でイタリックにはしないものは，予めコマンドが用意されていたりする．
例えば，三角関数
$$
f(\theta) = \sin(\theta) \\
g(\theta) = \cos(\theta) \\
h(\theta) = \tan(\theta) \\
$$
，　指数関数
$$
x(t) = x_0 \exp(a t)
$$
など．

## 数式を（等号などで）揃える
揃えたい文字を`&`で囲む（例．`&=&`）．
`eqnarray`環境や`align`環境で使える．数式をきれいに揃えたいときに試してみよう．

$$
\begin{eqnarray}
\bar{X} + n_{t+1} &=& f\left(\bar{X}+n_t\right)\\
&=& f\left(\bar{X}\right) + \frac{df}{dX}n_t + \frac{1}{2}\frac{d^2f}{dX^2}n_t^2+ \cdots
\end{eqnarray}
$$
```



LaTeXを使った数式の表示（2）

ギリシャ文字

\名前になっている．大文字は最初のアルファベットが大文字．

$\alpha, \beta, \gamma, \cdots$や$\Delta, \Phi$など．

改行

\\のようにバックスラッシュを２つ重ねる．

$$f(x) = x^2$$
$$g(x) = x^2 + 2x + 1$$

ドット，三点リーダ

ドット積でも使ったが，\cdotで中黒を表示できる．また三点リーダが各方向に対して使える．

- 水平 \cdots ⋯
- 垂直 \vdots ⋮
- 斜め \ddots ⋱

また，水平方向の三点リーダの底面バージョン…は\ldotsで表現できる．

イタリックにしない関数

名前のついている関数など慣例でイタリックにはしないものは，予めコマンドが用意されていたりする．例えば，三角関数

$$f(\theta) = \sin(\theta)$$
$$g(\theta) = \cos(\theta)$$
$$h(\theta) = \tan(\theta)$$

，　指数関数

$$x(t) = x_0 \exp(at)$$

など．

数式を（等号などで）揃える

揃えたい文字を&で囲む（例．&=&）．eqnarray環境やalign環境で使える．数式をきれいに揃えたいときに試してみよう．

$$\bar{X} + n_{t+1} = f\left(\bar{X} + n_t\right)$$
$$= f\left(\bar{X}\right) + \frac{df}{dX}n_t + \frac{1}{2}\frac{d^2f}{dX^2}n_t^2 + \cdots$$

Colabでの数式等の入力 LaTeX（3）

```
# $\LaTeX$を使った数式の表示（3）
## 括弧の大きさを自動で調整する
単純に`()`で囲んだだけだと，縦方向が不揃いになることがある.

$$
\frac{dx}{dt} = r (1-\frac{x}{K})x
$$

`\left( ... \right)`とすることで，高さが自動で調整される．他の種類の括弧（`[]`, `||`, `\{\}`）にも使える.

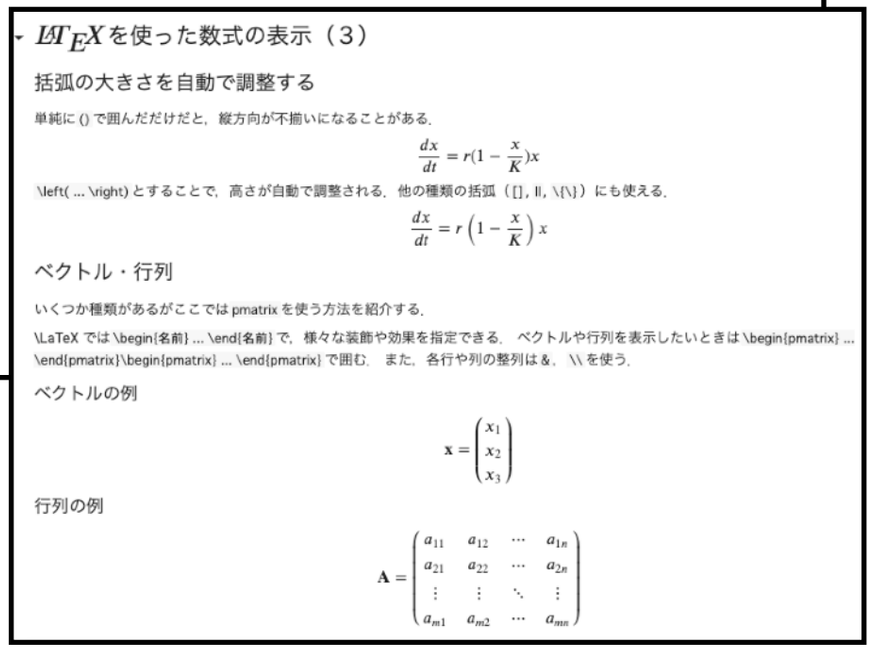
$$
\frac{dx}{dt} = r \left(1-\frac{x}{K}\right)x
$$


## ベクトル・行列
いくつか種類があるがここでは`pmatrix`を使う方法を紹介する.

\LaTeX では`\begin{名前} ... \end{名前}`で，様々な装飾や効果を指定できる.
ベクトルや行列を表示したいときは`\begin{pmatrix} ... \end{pmatrix}`で囲む.
また，各行や列の整列は`&`，`\\`を使う.

### ベクトルの例
$$
\mathbf{x} = \begin{pmatrix}
x_1\\
x_2\\
x_3
\end{pmatrix}
$$

### 行列の例
$$
\mathbf{A} = \begin{pmatrix}
a_{11} & a_{12} & \cdots & a_{1n} \\
a_{21} & a_{22} & \cdots & a_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
a_{m1} & a_{m2} & \cdots & a_{mn}
\end{pmatrix}
$$
```



他の数式の表現の仕方は調べながら使えるようになっていくと良い.
LaTeX全般についても知りたい人は調べてみよう.

- 参考
  - LaTeXコマンド集 http://www.latex-cmd.com/
  - LaTeXコマンドシート一覧 http://www002.upp.so-net.ne.jp/latex/index.html
  - LaTeX入門| TeX Wiki https://texwiki.texjp.org/?LaTeX%E5%85%A5%E9%96%80

本日の課題 ノーマル

1. 指数増殖，ロジスティック成長モデルを解析的に解け.
2. ロジスティック成長モデルについて１．で求めた解析解とオイラー法により近似した数値解を一つの図にプロットせよ．（その際のパラメータや初期値も示すこと）
3. 2. の図について，時間方向の刻み幅⊿tを様々に変化させ，その影響を考察せよ.
4. 質問，意見，要望等をどうぞ.

課題をノートブック（.ipynbファイル）にまとめて，Moodleにて提出すること

ファイル名は[回数，01~15]_[難易度，ノーマル nかハード h].ipynb．例．03_n.ipynb

本日の課題 ハード

1. カナダのモミの森林における害虫として知られているトウヒノシントメハマキ（spruce budworm）という蛾の幼虫がいる．この蛾は，個体数密度が一定以上になると殺虫剤などによる駆除をおこなっても減少させるのが難しい．逆に，個体数密度がある程度下がるとしばらくこうした大量発生はおこらない．このようなダイナミクスを表現するためのモデルとしてLudwig et al. (1978)では以下のモデルが提案された：

$$\frac{dx}{dt} = rx\left(1 - \frac{x}{K}\right)x - \beta\frac{x^2}{\alpha^2 + x^2}$$

   このモデルの平衡点やその局所安定性解析，数値的なシミュレーションなどから，どうしてトウヒノシントメハマキのダイナミクスをある程度再現できるのかを考察せよ．
   以下の教科書が参考になる；巌佐（1998），Murray (2007).

- Ludwig, D., Jones, D., Holling, C. (1978). Qualitative Analysis of Insect Outbreak Systems: The Spruce Budworm and Forest The Journal of Animal Ecology 47(1), 315. https://dx.doi.org/10.2307/3939
- 巌佐庸 (1998). 数理生物学入門 生物社会のダイナミックスを探る．共立出版,
- Murray, J. (2007). Mathematical biology: I. An introduction. Springer, New York, NY

課題をノートブック（.ipynbファイル）にまとめて，Moodleにて提出すること
ファイル名は[回数，01~15]_[難易度，ノーマル nかハード h].ipynb．例．04_h.ipynb

次回予告

第５回：ロトカ-ボルテラ モデル

６月１５日

復習推奨

- ロトカ-ボルテラ モデル
- 力学系の局所安定性解析