生態模擬: 以C語言為例

Class 06 (2018/04/19)

Simple numerical calculations

6.1 How to use standard output for saving data to a file
6.2 How to use R for graphics
6.3 Numerical solution of a difference equation (& bifurcation)
6.4 Explicit Euler method for ODE
6.5 4-th order Runge-Kutta method for ODE
6.6 Application to a prey-predator model

Takeshi Miki

三木　健（海洋研究所）

6.1 How to use standard output for saving data to a file

You can use 'output redirection' for redirecting output to a file.

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

#define STEP 0.1
int main(void)
{
        int j, k;
        double x, y;

        for(j = 0; j <= 100; j++){
                x = j*STEP;
                printf("%lf\t%lf\n", x, sin(x));
        }
        return 0;
}
```

When you use mathematical functions, i) you need to include the math library and ii) add option –lm when compiling via either:

> gcc –lm ***.c -o ***.out

> gcc ***.c –lm –o ***.out

> gcc ***.c –o ***.out -lm

At Terminal, you need to put the following command.

```
miki$miki-VB-ubuntu:~$ ./graph.out > plot01.dat↵

miki$miki-VB-ubuntu:~$  less plot01.dat↵
```

6.2 How to use R for graphics: open plot_test.R

You can use gnuplot for graphical presentation of data.

At the editor window of Rstudio, you can load and check the result from the program

```
#load the data
test<-read.table("./plot01.dat")
#see the data
View(test)
```

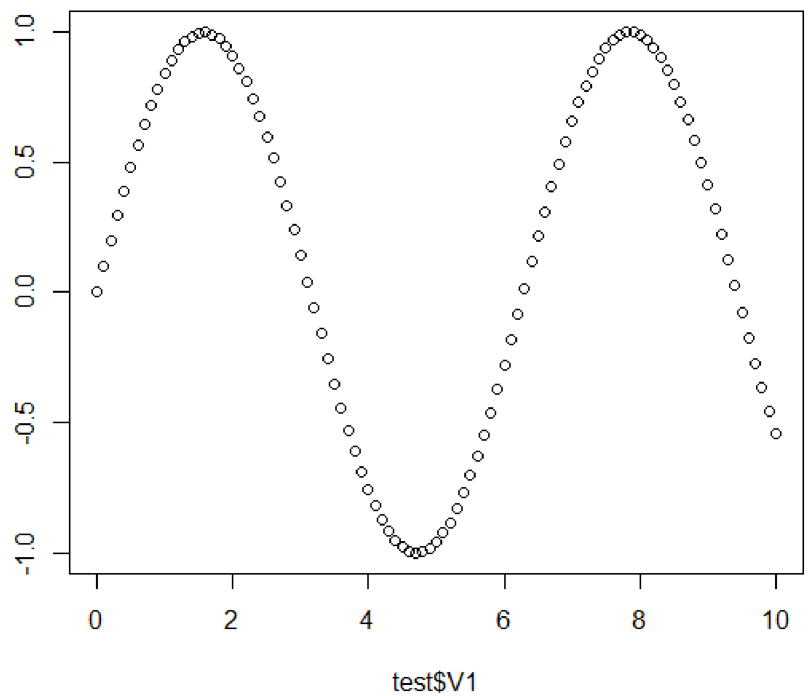
	V1	V2
1	0.0	0.000000
2	0.1	0.099833
3	0.2	0.198669
4	0.3	0.295520
5	0.4	0.389418
6	0.5	0.479426
7	0.6	0.564642
8	0.7	0.644218
9	0.8	0.717356
10	0.9	0.783327
11	1.0	0.841471
12	1.1	0.891207
13	1.2	0.932039
14	1.3	0.963558
15	1.4	0.985450
16	1.5	[illegible]

6.2 How to use R for graphics:

Simplest 2D scatterplot

With the following script, you can plot the result.

```
#simplest scatter plot
plot(test$V1,test$V2)
```

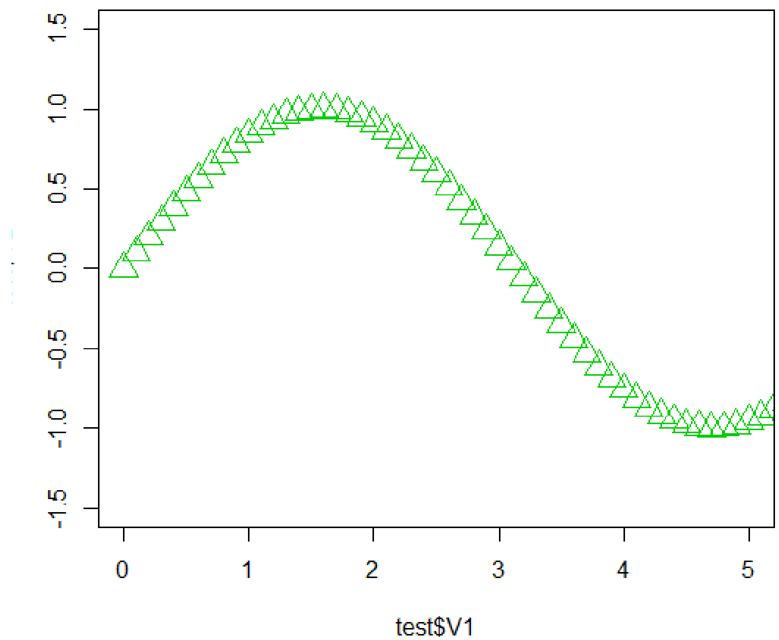


6.2 How to use R for graphics:

Some options

You can change the style of the plot with various options.

```
#change the options
plot(test$V1, test$V2, xlim=c(0,5),ylim=c(-1.5, 1.5), cex=2, pch=2, col=3)
```

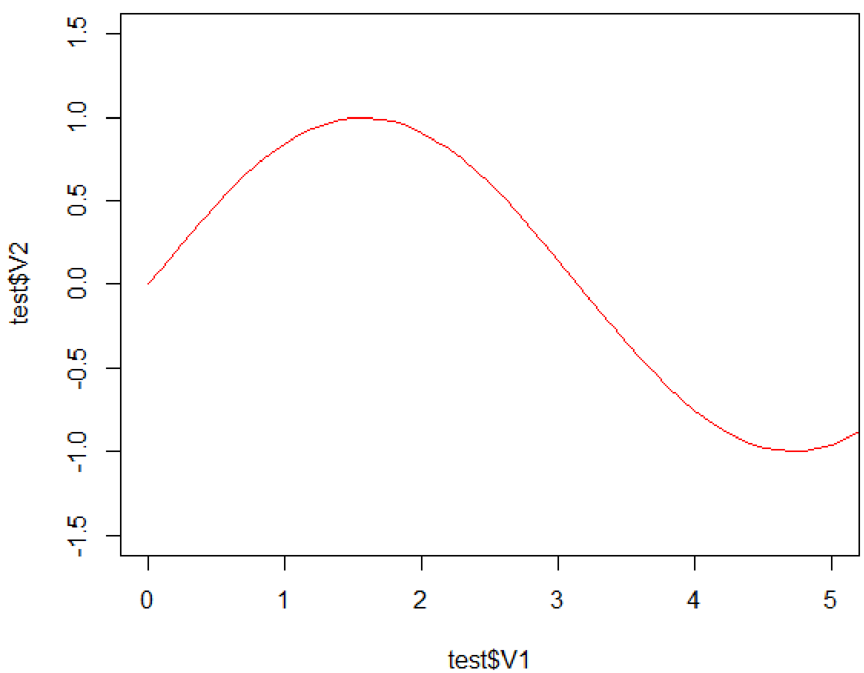


6.2 How to use R for graphics:

Some options

You can change the style of the plot with various options.

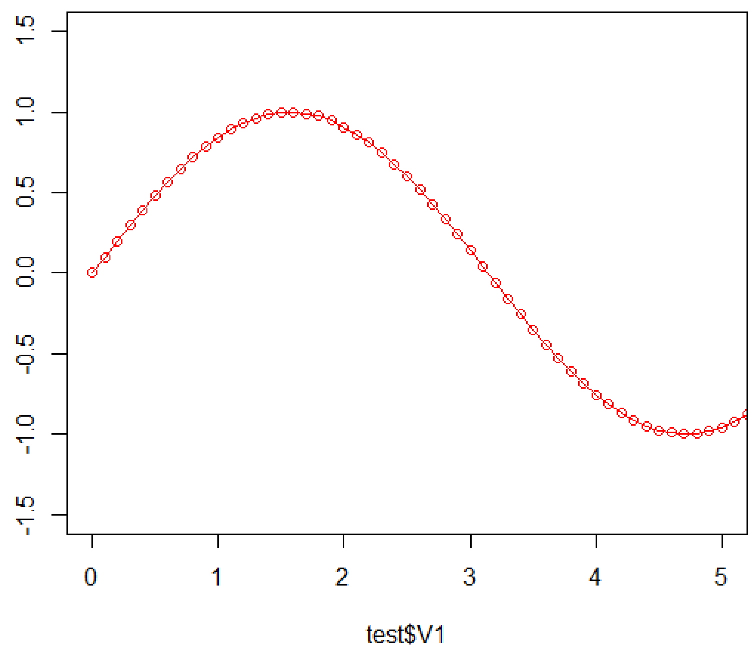
```
#lineplot
plot(test$V1, test$V2, xlim=c(0,5),ylim=c(-1.5, 1.5), type=‘l',col=2)
```



Some options

You can change the style of the plot with various options.

```
#linepoint plot
plot(test$V1, test$V2, xlim=c(0,5),ylim=c(-1.5, 1.5), type='o',col=2)
```

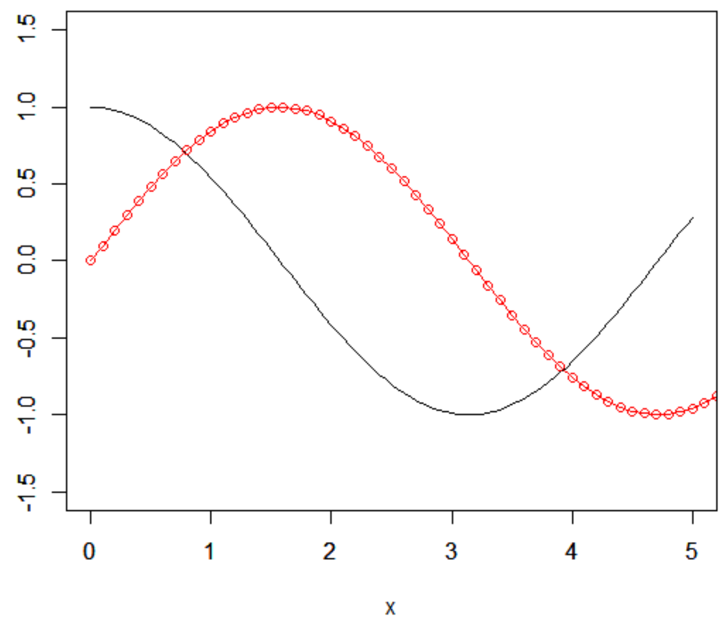


6.2 How to use R for graphics:

Some options

Two plots can be overdrawn.

```
#overdraw
plot(cos, 0,5, xlim=c(0,5), ylim=c(-1.5,1.5))
par(new=T)
plot(test$V1, test$V2, xlim=c(0,5),ylim=c(-1.5, 1.5), type='o',col=2, xlab='',ylab='')
```

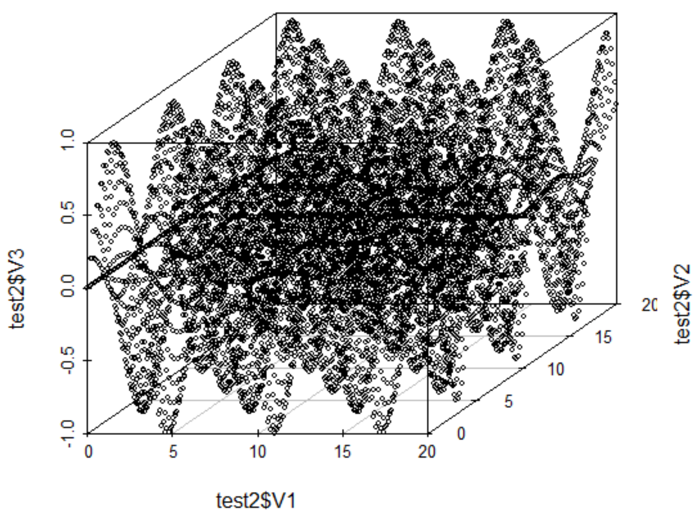


6.2 How to use R for graphics:

Some options

3D scatterplot is also possible with library::scatterplot3d

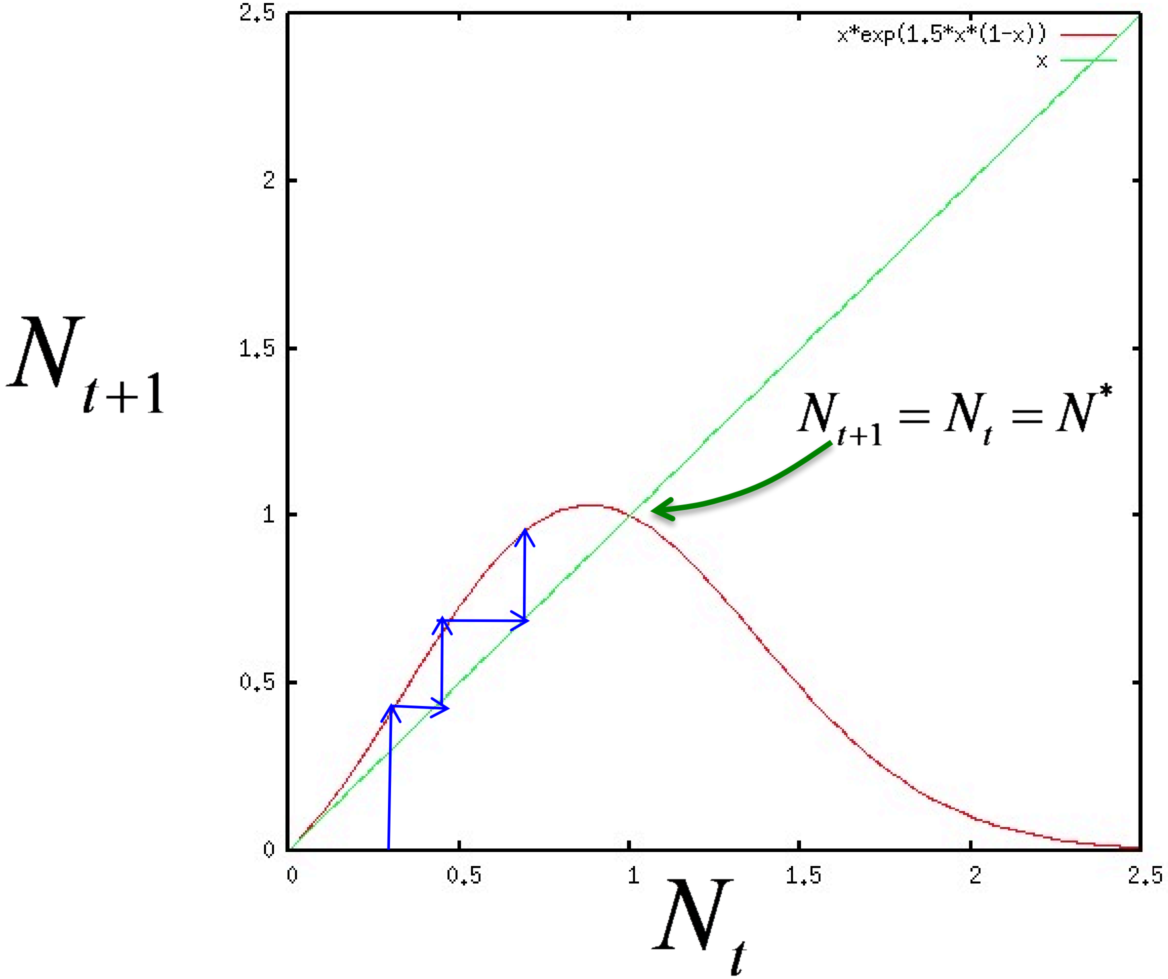
```
#3Dplot
test2<-read.table("./plot02.dat")
View(test2)
library(scatterplot3d)
scatterplot3d(test2$V1,test2$V2,test2$V3,cex.symbols=0.5)
```



6.3 Numerical solution of a difference equation (& bifurcation)

Let's consider the following simple population dynamics (Ricker Map).

$$N_{t+1} = N_t \exp\left[r \cdot (1 - N_t)\right] \quad N_0 = 0.1$$



6.3 Numerical solution of a difference equation (& bifurcation)

Let's consider the following simple population dynamics (Ricker Map).

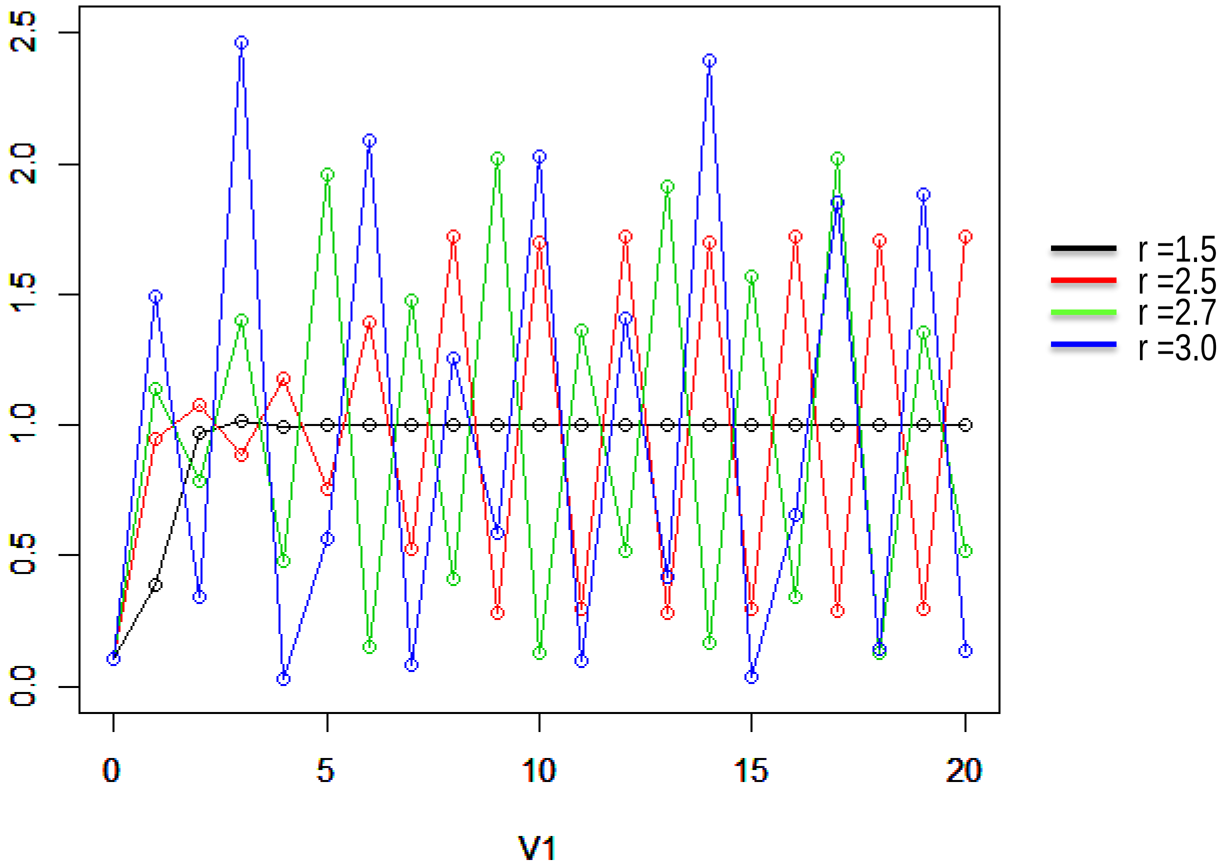
$$N_{t+1} = N_t \exp\left[r \cdot (1 - N_t)\right] \quad N_0 = 0.1$$

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

int main(void)
{
        int j;
        int end_step = 20;
        double x;
        double r = 1.5;

        x = 0.1;        //initial condition
        for(j = 0; j <= end_step; j++){
            x = x*exp(r*(1.0 – x));
            printf(”%d\t%lf\n", j, x);
        }
        return 0;
}
```

6.3 Numerical solution of a difference equation (& bifurcation)

Let's consider the following simple population dynamics (Ricker Map).

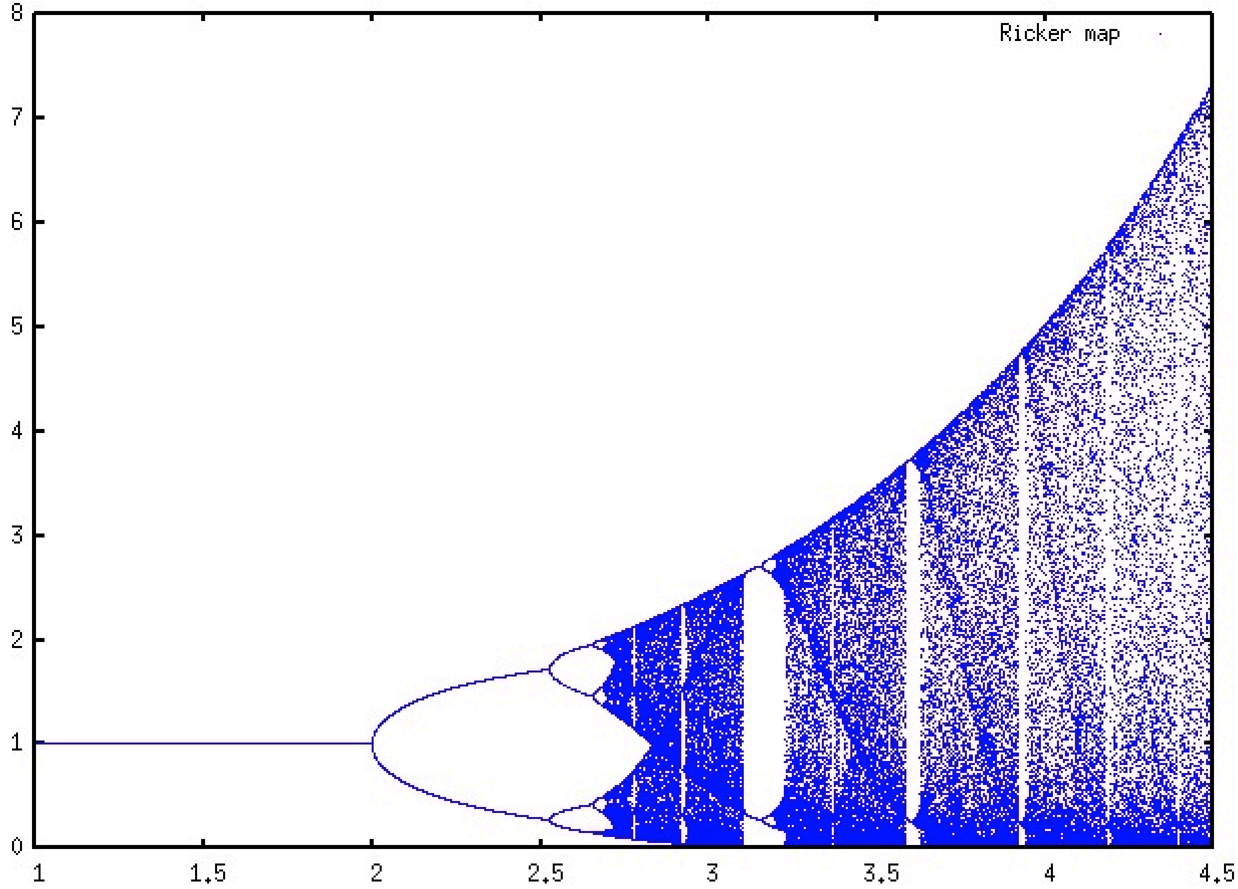
$$N_{t+1} = N_t \exp\left[r \cdot (1 - N_t)\right] \quad N_0 = 0.1$$



6.3 Numerical solution of a difference equation (& bifurcation)

We can easily obtain the bifurcation diagram (Ricker Map).

$$N_{t+1} = N_t \exp\left[r \cdot (1 - N_t)\right] \quad 1 \le r \le 4.5$$



N_t

r

Homework 1 !!

6.4 Explicit Euler method for ODE

Let's consider a one-dimensional ordinary differential equation (1D ODE)

$$\begin{cases} \dfrac{dN(t)}{dt} = f\left(N(t), t\right) \\ N(0) = N_0 \end{cases}$$

The explicit Euler method approximates N as n.

$$\begin{cases} \dfrac{n(t_1 + h) - n(t_1)}{h} = f\left(n(t_1), t_1\right) \\ n(0) = N_0 \end{cases}$$

6.4 Explicit Euler method for ODE

The explicit Euler method (explicit Euler discretization) is:

$$\begin{cases} \dfrac{n(t_1+h)-n(t_1)}{h} = f\left(n(t_1),t_1\right) \\ n(0)=N_0 \end{cases} \qquad (1)$$

From eqn.1,

$$\boxed{n(t_1+h)=n(t_1)+h\cdot f\left(n(t_1),t_1\right)} \qquad (2)$$

The true solution N(t) satisfies the following condition,

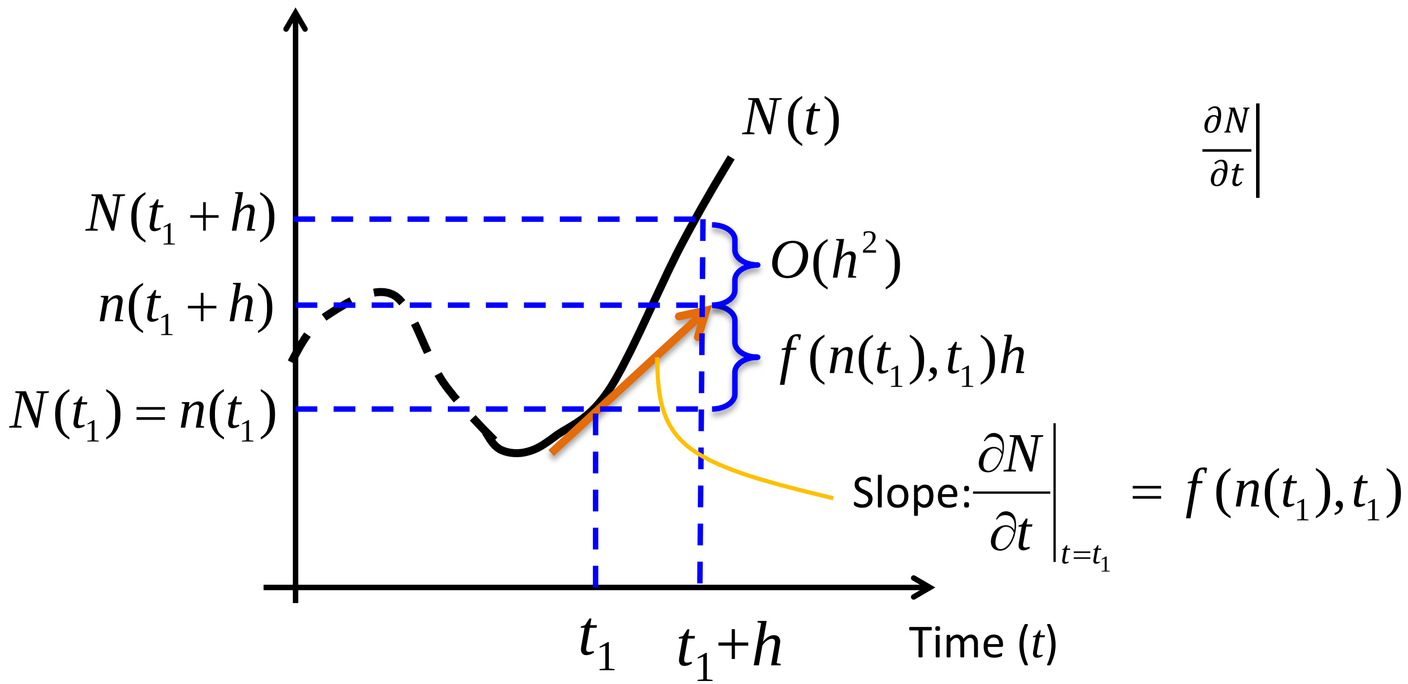
$$N(t_1+h)=N(t_1)+h\cdot\left.\frac{\partial N}{\partial t}\right|_{t=t_1}+\frac{1}{2!}h^2\cdot\left.\frac{\partial^2 N}{\partial t^2}\right|_{t=t_1}+O(h^3) \qquad (3)$$

6.4 Explicit Euler method for ODE

From Eqns. 2 and 3,

$$N(t_1+h)-n(t_1+h)=O(h^{1+1})$$

We say the explicit Euler method has first order accuracy (= local accuracy).



$\left.\frac{\partial N}{\partial t}\right|$

6.4 Explicit Euler method for ODE

From Eqns. 2 and 3,

$$N(t_1 + h) - n(t_1 + h) = O(h^{1+1})$$

We say the explicit Euler method has first order accuracy (= local accuracy).

Stability behavior of the Euler method

Consider the following linear ODE. $$\frac{dY}{dt} = \lambda \cdot Y$$

If we use the explicit Euler method, $$y(i \cdot h) = (1 + h\lambda)^i Y_0$$

The numerical solution is decaying (= stable) only when,

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

6.4 Explicit Euler method for ODE

cf) Implicit Euler method is…

$$\begin{cases} \dfrac{n(t_1) - n(t_1 - h)}{h} = f\left(n(t_1), t_1\right) \\ n(0) = N_0 \end{cases}$$

Stability behavior of the Euler method

Consider the following linear ODE. $$\frac{dY}{dt} = \lambda \cdot Y$$

If we use the implicit Euler method, $$y(i \cdot h) = \left(\frac{1}{1 - h\lambda}\right)^i Y_0$$

The numerical solution is decaying (= stable) only when,

$$\left|1 + h\lambda\right| > 1$$

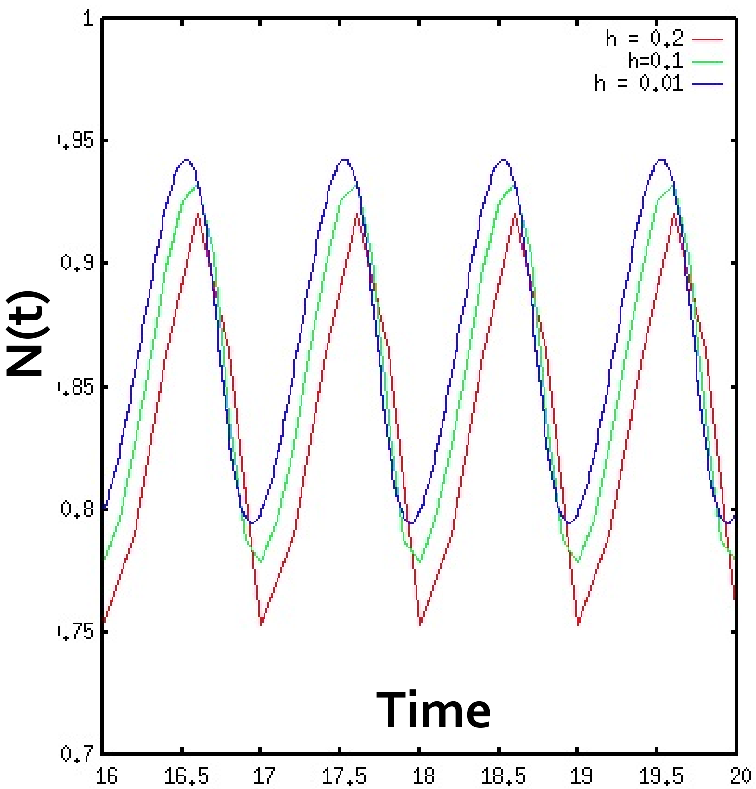
6.4 Explicit Euler method for ODE

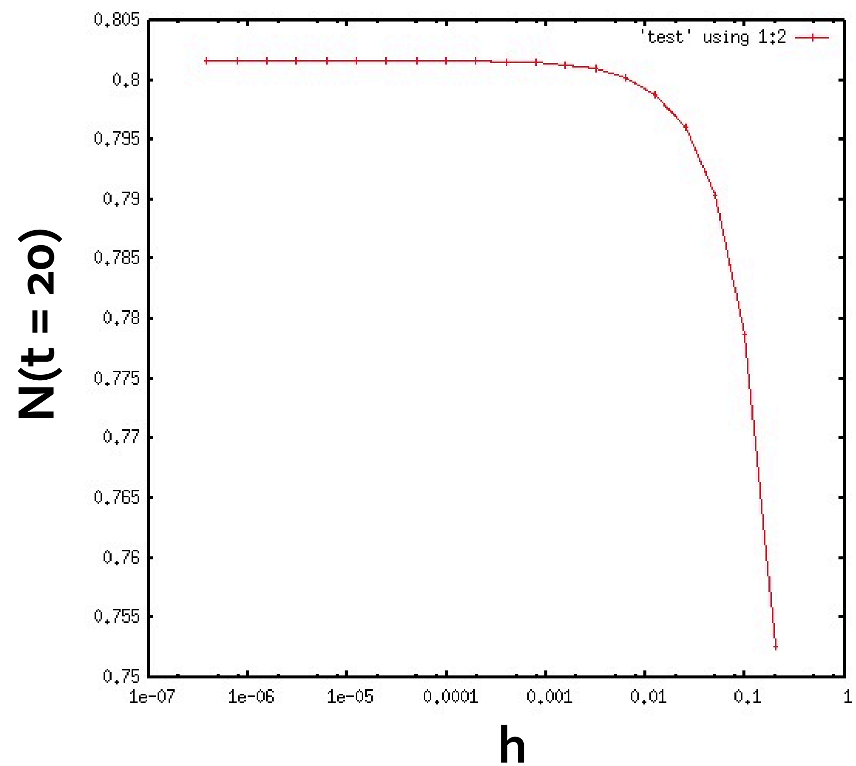
Use the explicit Euler method.

$$n(t_1 + h) = n(t_1) + h \cdot f\left(n(t_1), t_1\right)$$

$$\begin{cases} \dfrac{dN(t)}{dt} = 1.0 \cdot N(t) \cdot \left(1.0 - \dfrac{N(t)}{[1.0 + 0.5 \cdot \sin(2\pi t)]}\right) \\ N(0) = 0.1 \end{cases}$$

圓周率 M_PI





6.5 4-th order Runge-Kutta method for ODE

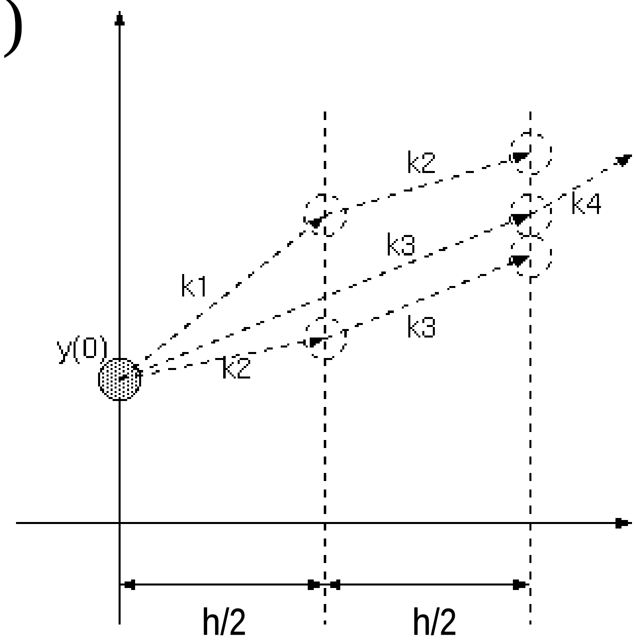
Let's consider a one-dimensional ordinary differential equation (1D ODE)

$$\begin{cases} \dfrac{dN(t)}{dt} = f\left(N(t), t\right) \\ N(0) = N_0 \end{cases}$$

The 4th order (explicit) Runge-Kutta method is..

$$n(t_1 + h) = n(t_1) + h \cdot \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

$$\textbf{with} \begin{cases} k_1 = f(n(t_1), t_1) \\ k_2 = f\left(n(t_1) + \dfrac{h}{2}k_1, t_1 + \dfrac{h}{2}\right) \\ k_3 = f\left(n(t_1) + \dfrac{h}{2}k_2, t_1 + \dfrac{h}{2}\right) \\ k_4 = f(n(t_1) + hk_3, t_1 + h) \end{cases}$$



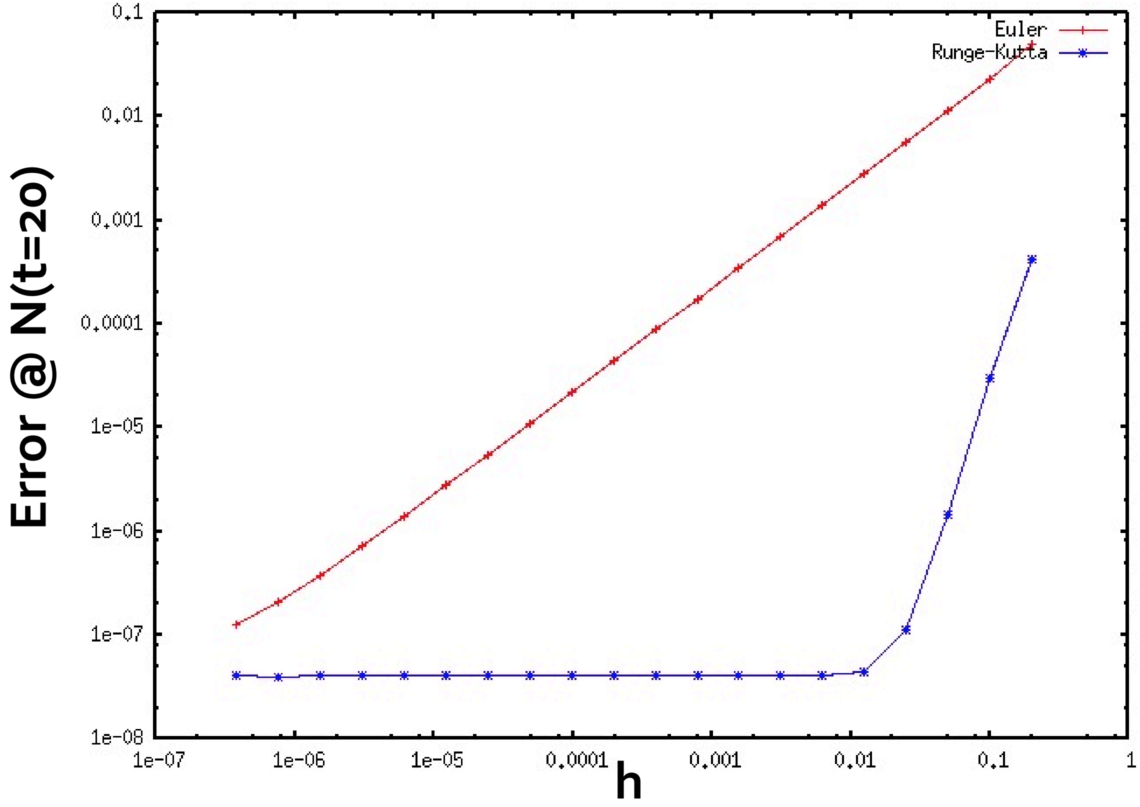
You can mathematically show the difference with the true solution as follows.

$$N(t_1 + h) - n(t_1 + h) = O(h^{4+1})$$

6.5 4-th order Runge-Kutta method for ODE

Compare the explicit Euler method and the Runge-Kutta method.

$$\begin{cases} \dfrac{dN(t)}{dt} = 1.0 \cdot N(t) \cdot \left(1.0 - \dfrac{N(t)}{[1.0 + 0.5 \cdot \sin(2\pi t)]}\right) \\ N(0) = 0.1 \end{cases}$$



6.6 Application to a prey-predator model

Numerically solve the following equations with the Runge-Kutta method.

$$
\begin{cases}
\dfrac{dx}{dt} = rx\left(1 - \dfrac{x}{K}\right) - a\dfrac{xy}{1 + H_t ax} \\[2ex]
\dfrac{dy}{dt} = b\dfrac{xy}{1 + H_t ax} - my \\[6ex]
x(0) = 0.1, y(0) = 0.1
\end{cases}
$$

$$
r = 1.0, a = 1.0, b = 0.5, H_t = 1.0, m = 0.1
$$

6.6 4-th order Runge-Kutta method for ODE

Runge-Kutta method is the same for the multidimensional ODE.

$$\begin{cases} \dfrac{dN(t)}{dt} = f\big(N(t), t\big) \\ N(0) = N_0 \end{cases}$$

The 4th order (explicit) Runge-Kutta method is..

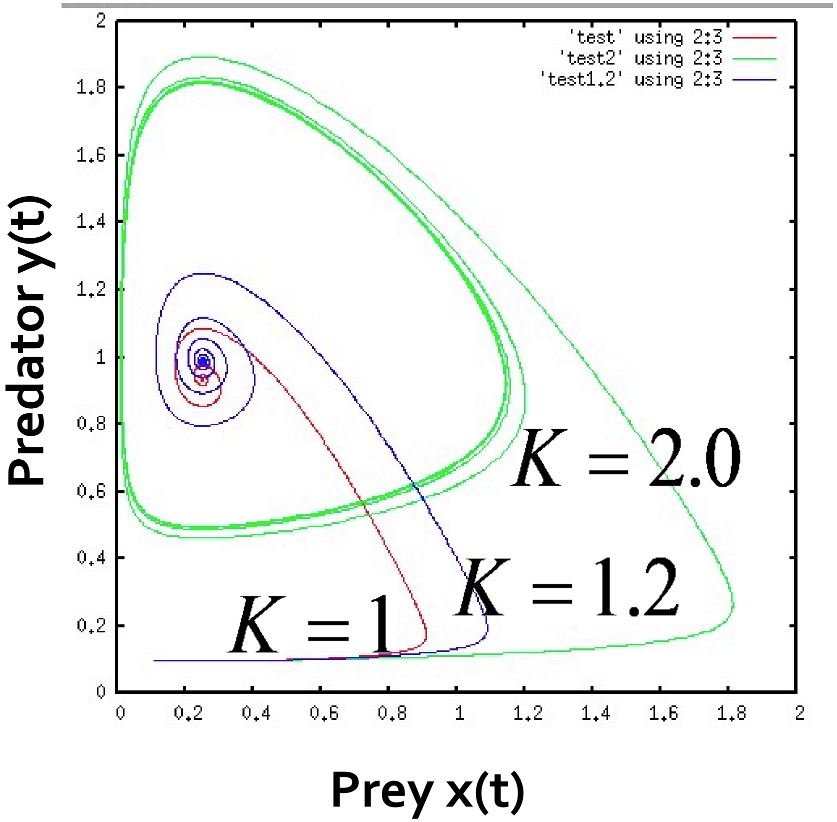
$$n_{i+1} = n_i + h \cdot \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

with

$$\begin{cases} k_1 = f(n_i, t_i) \\ k_2 = f\left(n_i + \dfrac{h}{2}k_1, t_i + \dfrac{h}{2}\right) \\ k_3 = f\left(n_i + \dfrac{h}{2}k_2, t_i + \dfrac{h}{2}\right) \\ k_4 = f(n_i + hk_3, t_i + h) \end{cases}$$

6.6 Application to a prey-predator model

Numerically solve the following equations with the Runge-Kutta method.



Homework 2 !!