The case: insular geckos on an Aegean islet

Group:
Christian Berrig
Amalia Bogri

Species:
Mediodactylus kotschyi

Location:
Exo Diavates, area: 20 sqm
Carrying capacity: 1000 individuals

K

Life history:
Longevity: 4.5 years
Sexual maturity: at 2 years
Clutch size: 2 eggs per female
Clutch frequency: 2 clutches per year
(Çiçek *et al.*, 2015; Schwarz *et al.*, 2020)

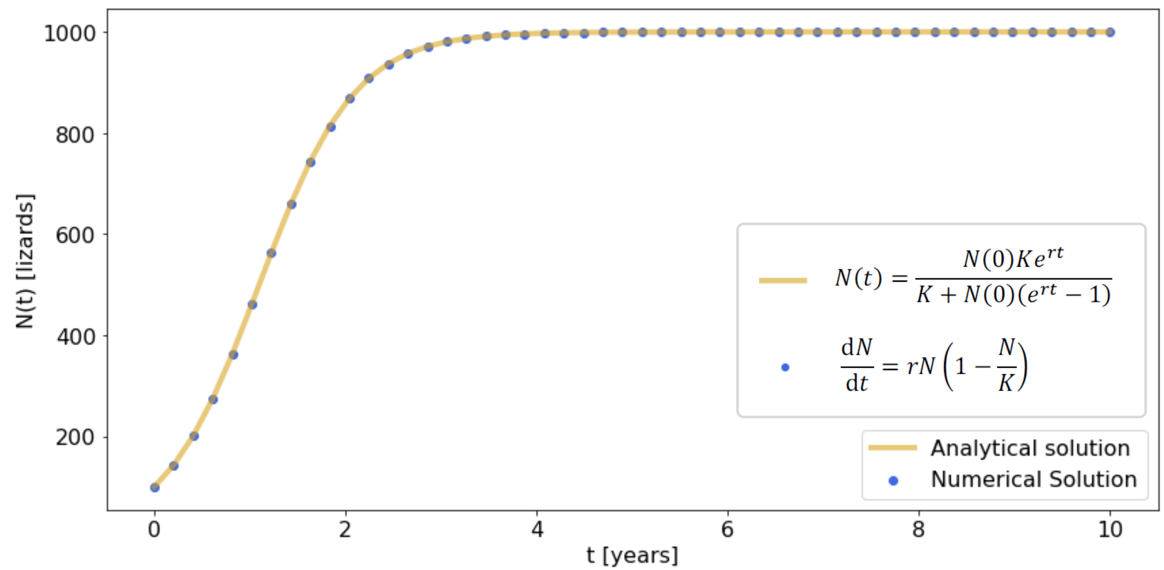
How many eggs? 2 eggs per gecko per year

Over how many years? 6 reproductive years

Birth rate: 2 eggs x 6 years / 8 years
Death rate: 1/8 years
Growth rate ~ 1.4

r

Parameters [units]:
K = 1000 [individuals]
N0 = 100 [individuals]
r ~ 1.4 [1/year]

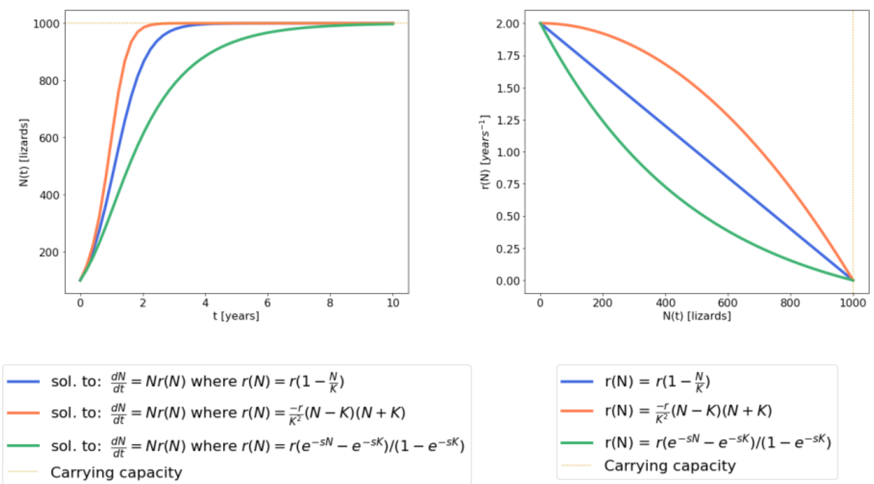


Additional topics

Density Dependence:

Parameters:
K = 1000, N0 = 100,
r : is a function of N, with arbitrary choice of equations to produce convex and concave r-N graph

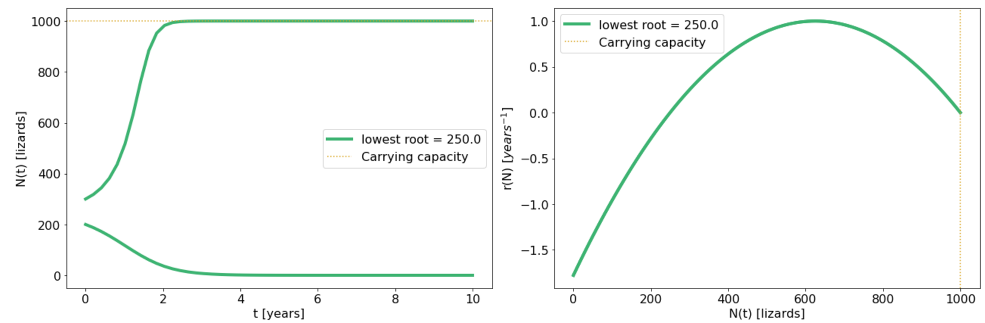
Time: 10 years, in 50 increments



Allee effect:

Parameters:
K = 1000, **N0 = 200 and 300**,
r : is a function of N, with N modelled as a 2^{nd} order polynomial.

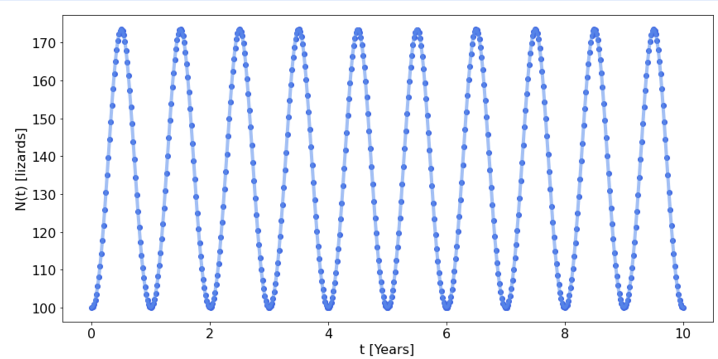
Notes:
If the population starts from less than 250 individuals, the growth rate will be negative, and the population will get extinct. Biologically, this could be explained from the difficulty to find mates if the population is sparsely distributed.



Seasonality:

Parameters:
K = 1000, N0 = 100,
r : dependent on sin(2π t)

Time: 10 years, in **500** increments.



* Numerical solutions shown for all graphs herein.

Extra details (for the teachers)

Maximum growth rate calculations:

How many eggs?
2 eggs x 2 clutches/year =
4 eggs per gecko pair/year =
2 eggs per one gecko/year=
2 geckos per gecko per year

Over how many years?
8 years (lifetime) – 2 years (sexual immaturity) =
6 reproductive years

Birth rate:
2 geckos per gecko x 6 *reproductive* years / 8 years *of life*

Death rate: 1/8 years *of life*

Growth rate (r): Birth rate – Death rate ~ 1.4

Carrying capacity:
We choose a K of 1000 individuals, based on the size of the islet (20 sqm) and the small size and cryptic nature of the lizards.

Initial population:
We chose to start our model with a small population of 100 individuals.

In the case of the Allee effect, we put the threshold of critical population size on 250 individuals, and we showcase the examples of N0=200 and N0=300.

Time:
We chose a time scale in years and we show the progress of the population for 10 years. For the numerical solutions of most ODEs, the time was split to 50 time-steps (increments).
In the case with the seasonality, however, we used 500 time-steps to achieve more accurate solutions.

Growth rate as function of population:
We chose arbitrary functions of r dependent on the population size to showcase the different scenarios.
However, we do not have any data to support which of the possible growth rates is correct for this lizard.
For example, the Allee effect should not be an issue, as lizards can move relatively fast and should be able to locate each other in such a small islet.

Growth rate as a function of time:
The seasonality scenario is quite realistic, as indeed this species of lizard only reproduces in the warm months of the year (i.e. April-September).
We should note, however, that the growth rate in our model is actually quite low, as a function of sin(2πt), and the population is thus constrained to very low numbers (max 200 individuals).
Therefore, more the r(t) function should be modified further.

References:
Çiçek K, Afsar M, Kumaş M, Ayaz D, Tok CV. 2015. Age, Growth and Longevity of Kotschy's Gecko, Mediodactylus kotschyi (Steindachner, 1870) (Reptilia, Gekkonidae) from Central Anatolia, Turkey. Acta zool. bulg., 67 (3): 399-404.

Schwarz R, Itescu Y, Antonis Antonopoulos A, GavriilidI IA, Tamar K, Pafilis P, Meiri S. 2020. Isolation and predation drive gecko life-history evolution on islands. Biological Journal of the Linnean Society, 129: 618–629.

Code (in Python)

```python
# Import modules:
from scipy.integrate import odeint
import matplotlib.pyplot as plt
import numpy as np

fs_label = 16 # This is for reasonable fontsize universally defined:
parameters = {'figure.titlesize': fs_label+6,
                'axes.labelsize': fs_label,
                'axes.titlesize': fs_label+4,
                'xtick.labelsize': fs_label,
                'ytick.labelsize': fs_label,
                'legend.fontsize': fs_label}
plt.rcParams.update(parameters)

# Logistic growth

# Parameters
life_years = 4.5 # total lifespan (average)
repr_years = life_years - 2 # lizards mature sexually at 2 years old
clutch_size = 2 # females lay two eggs per clutch
clutch_freq = 2 # females lay two clutches per year

# Birth rate: 2 eggs per lizard per reproductive year across lifetime
b = (clutch_size * clutch_freq) * repr_years / life_years
d = 1 / life_years # Death rate : one lizard lives for 4.5 years
r = b - d # Growth rate

K = 1000 # Carrying capacity (arbitrary choice)
N0 = 100 # Initial population (arbitrary choice)

t = np.linspace(0, 10, 50) # range for time

# Define the equations:
r_N = lambda N: r*(1 - N/K) # a function for logistic growth

def dN_dt(N, t, r):
    return N*r(N) # a ODE function for growth

def N_analytical(t, N0=N0, r=r, K=K):
    return (N0*K*np.exp(r*t))/(K+(N0*(np.exp(r*t)-1))) # the analytical solution equation

par = (r_N, )
ns = odeint(dN_dt, N0, t, args=par) # solve the ODE
an = N_analytical(t)

# Plot:
fig, ax = plt.subplots(figsize=(12,6), tight_layout=True)

ax.plot(t, an, color='goldenrod', linewidth=4, alpha = 0.6, label="Analytical solution")
ax.scatter(t, ns, color='royalblue', alpha = 1, label="Numerical Solution")

ax.set_xlabel("t [years]")
ax.set_ylabel("N(t) [lizards]")
ax.legend()
```

```python
# Density Dependence

# These 3 functions define resp. linear, concave and convex growthrate dependency as funct. of population N:
r_lin = lambda N: r*(1 - N/K)
r_concave = lambda N: -r/(K**2)*(N-K)*(N+K)
s=0.002 # arbitrary
r_convex = lambda N: r*(np.exp(-s*N)-np.exp(-s*K))/(1-np.exp(-s*K))
# these options are simply to make the automation of the plot a bit easier:
grs = [r_lin, r_concave, r_convex]
clr = ["royalblue", "coral", "mediumseagreen"]
lbl = ["$r (1-\\frac{N}{K})$", "$\\frac{-r}{K^{2}}(N-K)(N+K)$", "$r(e^{-sN}-e^{-sK})/(1-e^{-sK})$"]
iterators = [grs, clr, lbl]
# Plot:
fig, ax = plt.subplots(1, 2, figsize=(18,10), tight_layout=True, gridspec_kw={'width_ratios':[1,1]})
for f, c, l in zip(*iterators):
    par = (f, )
    ns = odeint(dN_dt, N0, t, args=par)
    ax[0].plot(t, ns, color=c, label="sol. to:  $\\frac{dN}{dt} = N r(N)$ where $r(N)=$"+l, linewidth=4)
    ax[1].plot(n, f(n), color=c, label="r(N) = "+l, linewidth=4)
ax[0].axhline(K, color='goldenrod', linestyle=":", label="Carrying capacity")
ax[1].axvline(K, color='goldenrod', linestyle=":", label="Carrying capacity")
ax[0].set_xlabel("t [years]")
ax[0].set_ylabel("N(t) [lizards]")
ax[1].set_xlabel("N(t) [lizards]")
ax[1].set_ylabel("r(N) [$years^{-1}$]")
for a in ax:
    a.legend(loc='upper center', bbox_to_anchor=(0.5, -0.2), borderaxespad=1, fontsize = 22)
```

```python
# Allee-effect
N0s = np.array([200, 300]) # this defines the initial conditions for solutions
# These growthrate dependent as funct. of population N modelled as 2. order polyn:r
r_polyn2 = lambda N, r1, r2: -((2*r)/((r1-r2)**2))*(N-r1)*(N-r2)
# Define roots for the 2. order plyn. Note that the "Allee-effect" will only happen if thr root is positive
r1s = [K/4]
# Plot:
for r1 in r1s:
    fig, ax = plt.subplots(1, 2, figsize=(18,6), tight_layout=True, gridspec_kw={'width_ratios':[1,1]})
    f = lambda N: r_polyn2(N, r1, K)
    l = f"lowest root = {r1}"
    par = (f, )
    for n0 in N0s:
        ns = odeint(dN_dt, n0, t, args=par)
        if n0 == N0s[0]:
            ax[0].plot(t, ns, color="mediumseagreen", label=l, linewidth=4)
            ax[1].plot(n, f(n), color="mediumseagreen", label=l, linewidth=4)
        else:
            ax[0].plot(t, ns, color="mediumseagreen", linewidth=4)
            ax[1].plot(n, f(n), color="mediumseagreen", linewidth=4)
    ax[0].axhline(K, color='goldenrod', linestyle=":", label="Carrying capacity")
    ax[1].axvline(K, color='goldenrod', linestyle=":", label="Carrying capacity")
    if r1 != r1s[0]:
            ax[0].axhline(r1, color="tab:cyan", linestyle=":", label="Critical Population")
            ax[1].axvline(r1, color="tab:cyan", linestyle=":", label="Critical Population")
    ax[0].set_xlabel("t [years]")
    ax[0].set_ylabel("N(t) [lizards]")
    ax[1].set_xlabel("N(t) [lizards]")
    ax[1].set_ylabel("r(N) [$years^{-1}$]")
    for a in ax:
        a.legend()
```

```python
# Seasonality
t = np.linspace(0, 10, 500) # range for time
def dN_dt(N, t, r, K):
    r_N_t = lambda N: (r*(np.sin(2*np.pi*t)))*(1 - N/K)
    return N*r_N_t(N)
par = (r, K)
ns = odeint(dN_dt, N0, t, args=par)
# Plot:
fig, ax = plt.subplots(figsize=(12,6), tight_layout=True)
ax.plot(t, ns, color='cornflowerblue', linewidth=4, alpha = 0.6)
ax.scatter(t, ns, color='royalblue', alpha = 1)
ax.set_xlabel("t [Years]")
ax.set_ylabel("N(t) [lizards]")
```