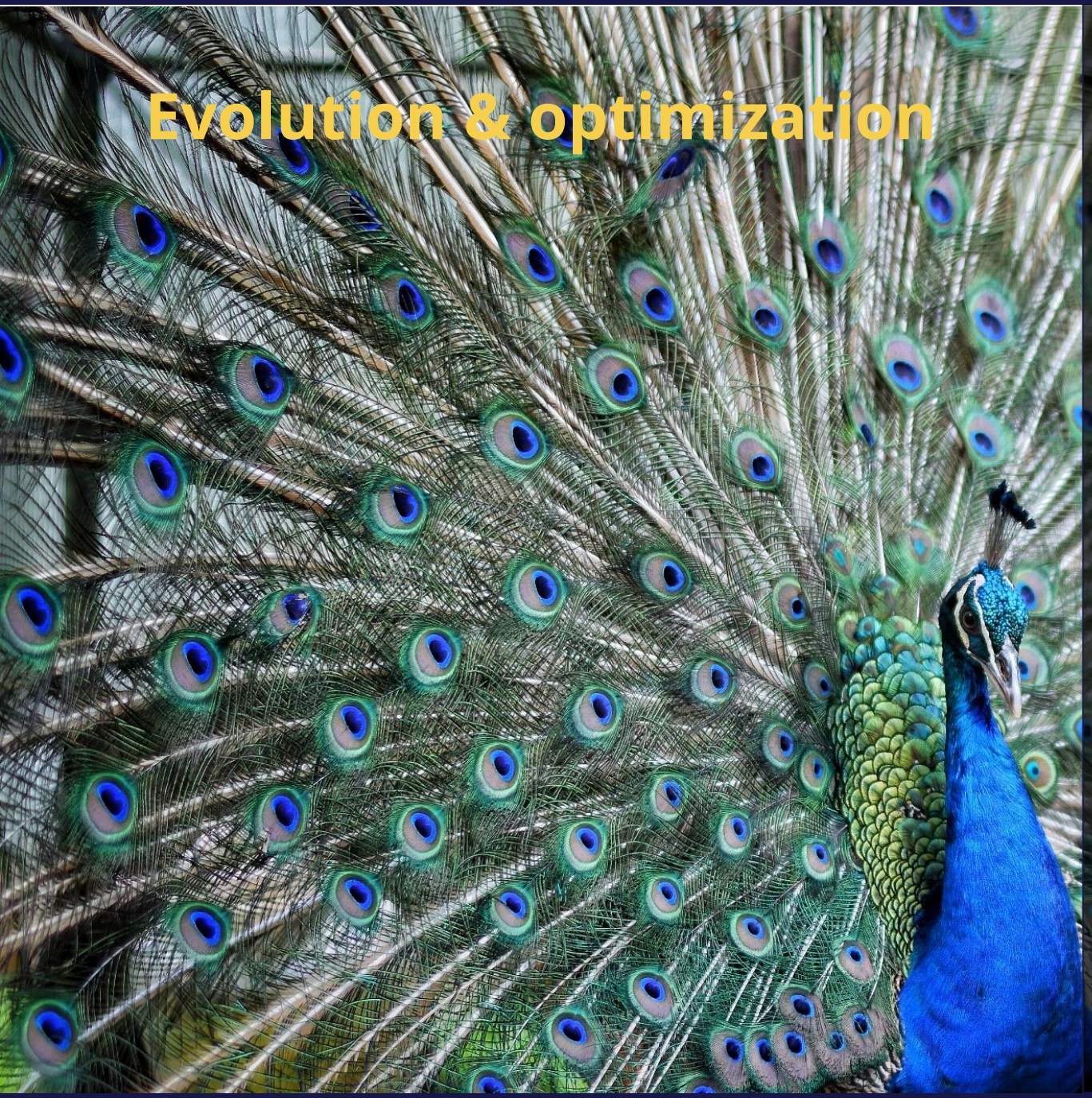
Group: Amalia Bogri, Christian Berrig & Jonas Bolduan

Evolution and optimization

Evolution & optimization



Females (peahens) prefer to mate with males (peacocks) that have the most elaborate trails,
i.e. the trails with the **most eye-spots**.

Trails with more eye-spots are also larger in diameter.

Thus, more eye-spots signify greater mating success (**gain**).
The eye-spots are the trait we will use.

The **costs** of carrying an elaborate trail are still unknown.

In this model, we assume that there is a
- metabolic cost (due to carrying the extra weight)
- immune suppression cost that leads to excess mortality (due to the increased testosterone) with more eye-spots

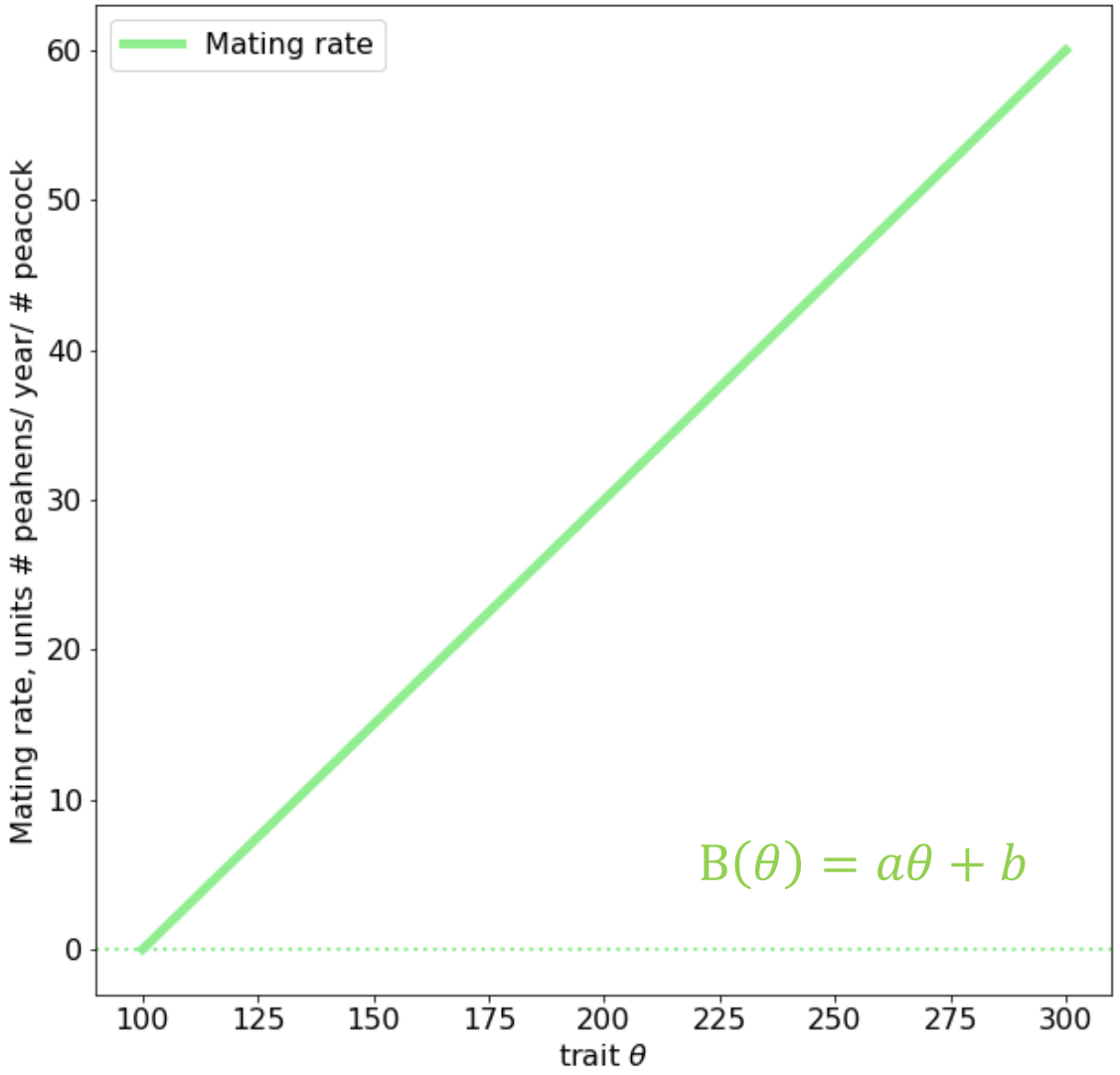
References:

Petrie et al., 1991, Petrie & Halliday 1994, Gadagkar 2003, Ros et al., 2009

Evolution & optimization: model parameters

Parameter	Unit	Value
Peahens, R	# peahens	$R_0 = 100$
Peacocks, N	# peacocks	
Trait (eye-spots), θ	-	100-300
Maximum mating, C_{max}	# peahens/ # peacock/ year	100
Mating efficiency, ε_R	-	0.5
Metabolism, M	1/year	0.2
Mating rate, $\mathrm{B}(\theta) = a\theta + b$	# peahens/ year/ # peacock	a=0.3, $b = -30$
Gains, $G(\theta) = \varepsilon_R \left(C_{max} \frac{\mathrm{B}(\theta)R}{\mathrm{B}(\theta)R + C_{max}} - M \right)$	1/year	
Background mortality, m_0	1/year	0.3
Immunosuppression mortality, m_p	1/year	0.2
Total mortality, $M_{tot}(\theta) = m_0 + m_p\theta$	1/year	
Growth rate, $r(\theta) = G(\theta) - M_{tot}(\theta)$	1/year	

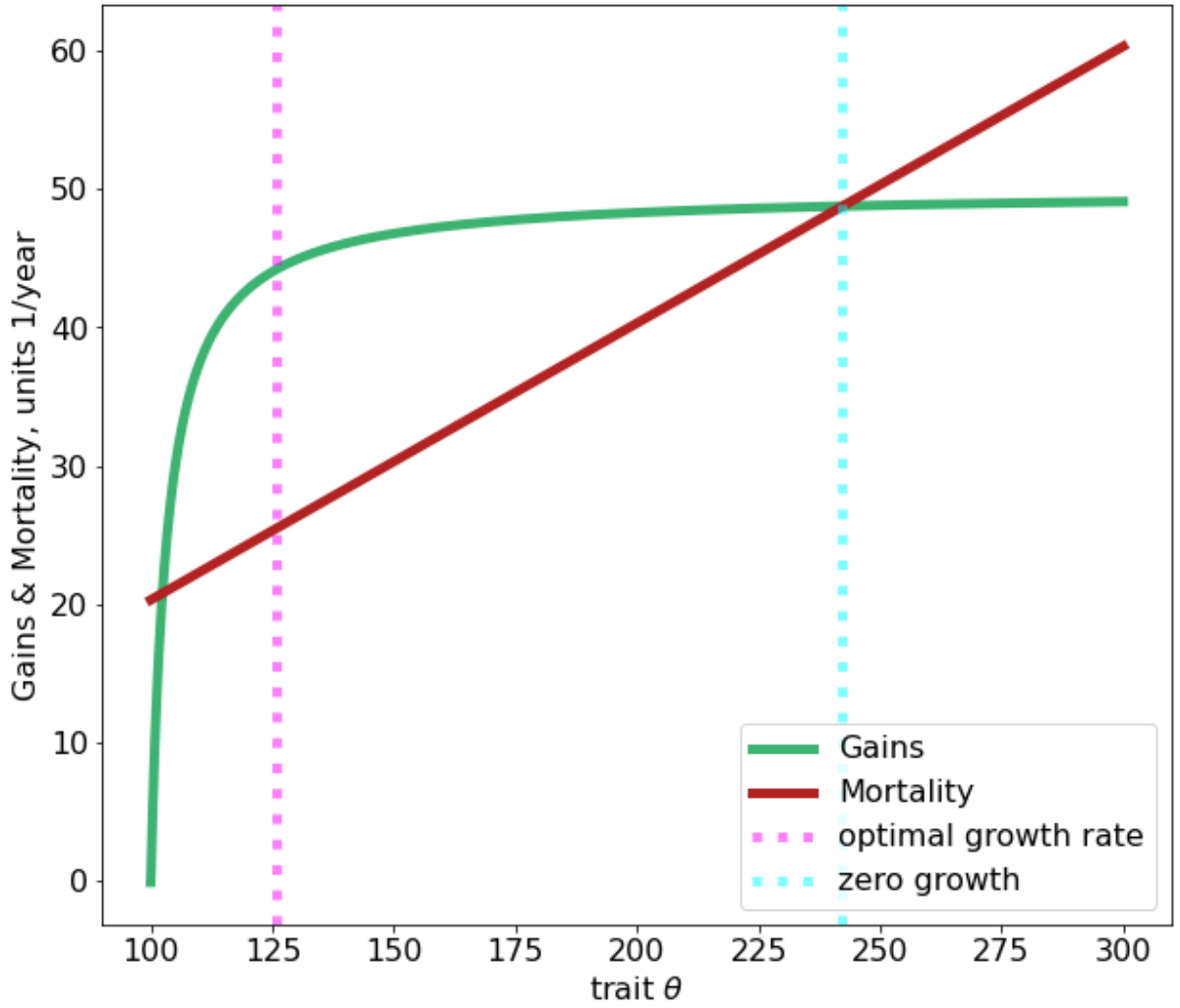
Evolution & optimization: plots



Mating rate* increases linearly with the number of eye-spots

*Similar to ‘clearance rate’

Evolution & optimization: plots



$$G(\theta) = \varepsilon_R \left(C_{max} \frac{\mathrm{B}(\theta) R}{\mathrm{B}(\theta) R + C_{max}} - M \right)$$

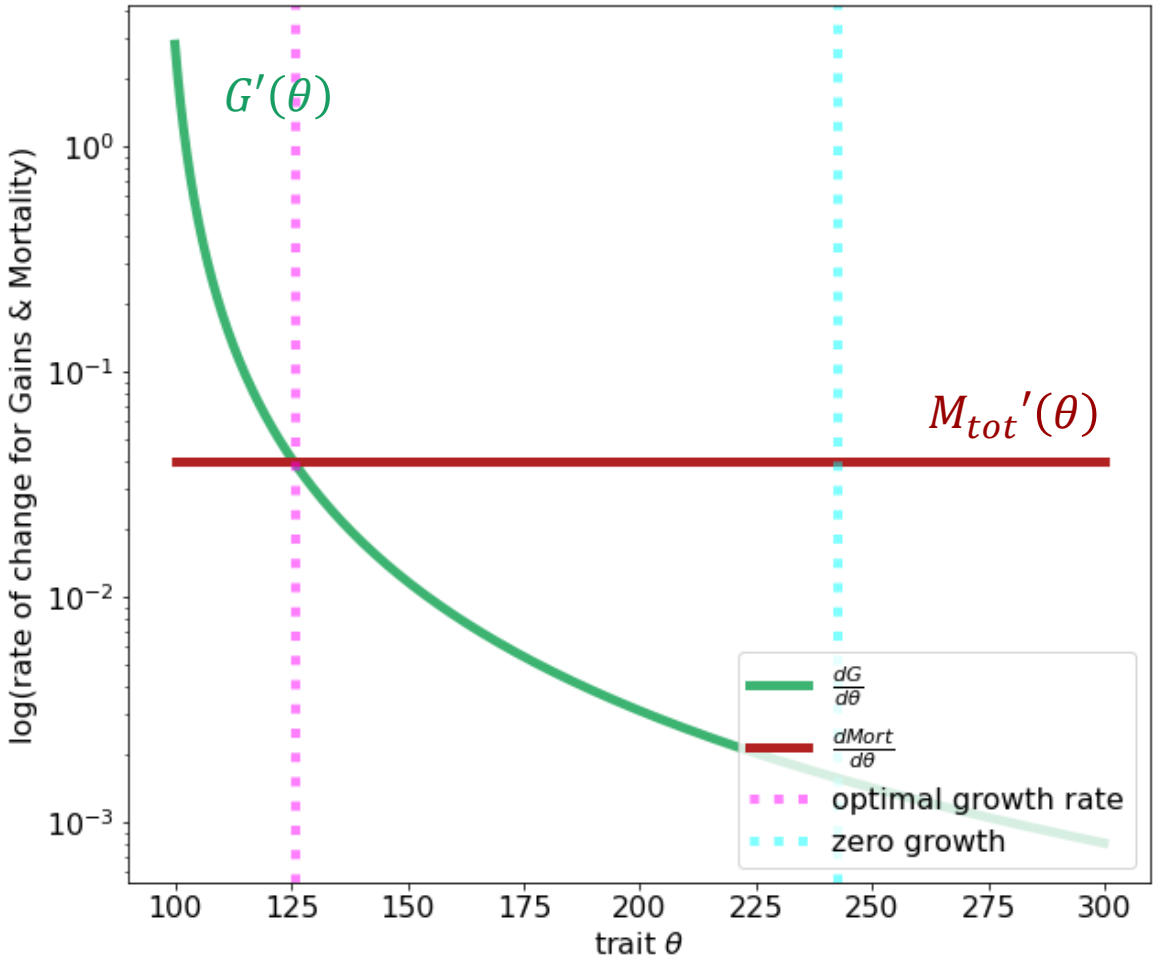
$$M_{tot}(\theta) = m_0 + m_p \theta$$

Total mortality increases with the number of eye-spots

Gains increase logistically up to a limit, at approximately 150 eye-spots.

(Optimal & zero growth calculated from growth rate r.)

Evolution & optimization: plots

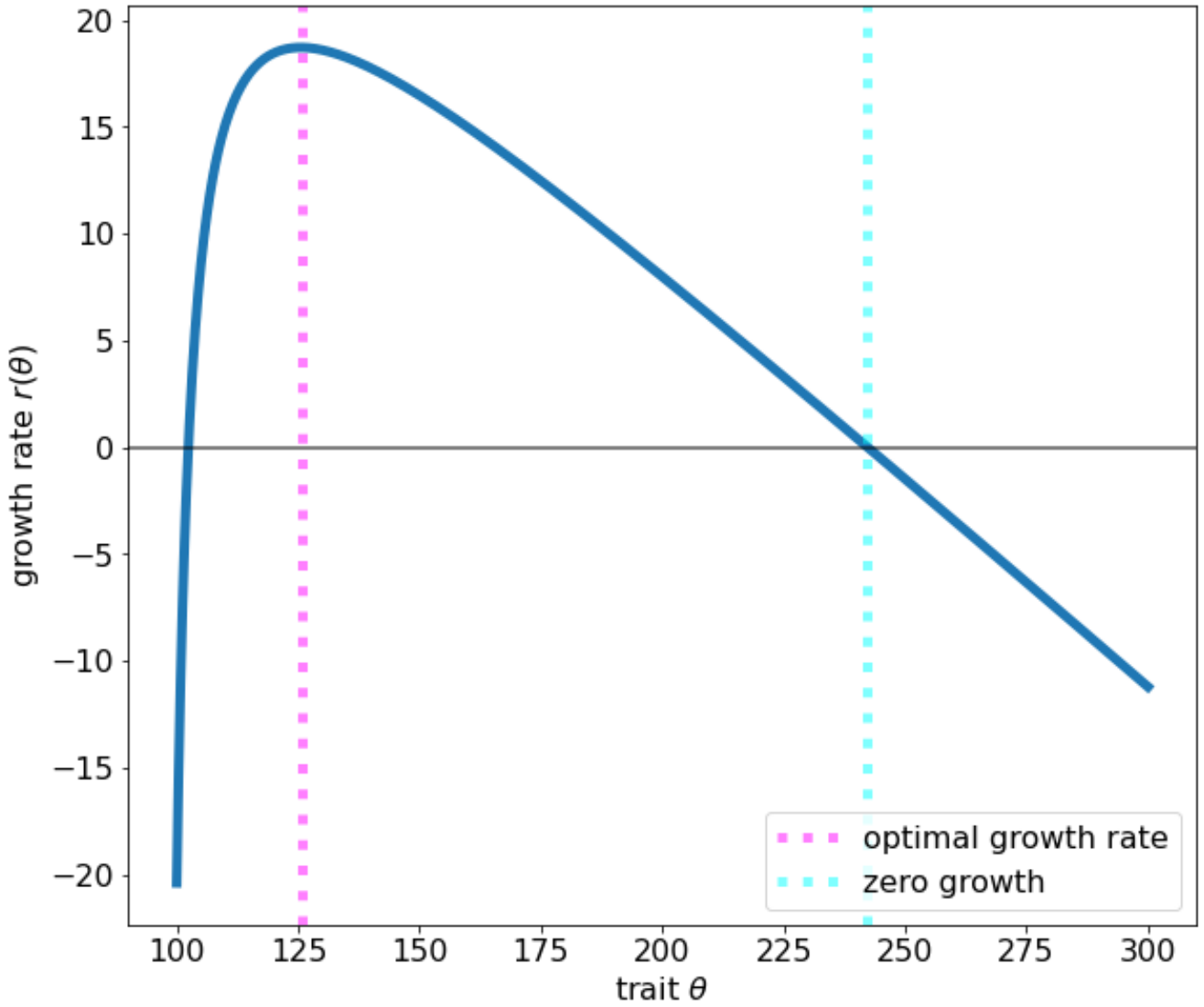


As expected, the rate of change of mortality is stable &

the rate of change of gains is decreasing (almost exponentially)

as the number of eye-spots increases

Evolution & optimization: plots



The growth rate of the peacocks reaches a maximum at ~125 eye-spots.

The growth rate becomes zero at ~240 eye-spots. This when the benefits of carrying the trait are equal to the costs.

At more than ~240 eye-spots, the metabolic & mortality costs of carrying the trait are greater than the mating benefit it confers.

Python code

```
# Import stuff:
from scipy.integrate import odeint
import matplotlib.pyplot as plt
import numpy as np
import scipy as sp
from sympy import Symbol
from sympy.solvers import solve

# This is for reasonable fontsize universally defined:
fs_label = 16
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,
                'lines.linewidth': 5
            }
plt.rcParams.update(parameters)

# Parameters
# Case: Peacocks

R = 100 # 'resource': peahens
c_max = 100 # 'max consumption': maximum matings(females) per year per peacock
M = 0.2 # metabolism 0.2
eps = 0.5 # 'efficiency': successfull matings leading to offspring
m0 = 0.3 # backgrond mortality
mp = 0.2 # mortality due to "predation"/'immune supression'
theta_list = np.linspace(100, 300, 1000)
b = -30
a=0.3

B = lambda theta: a*theta + b # 'clearance rate': Dimension: females/time/peacock, Unit:
sqkm/year/#peacocks
G = lambda theta: eps*(c_max*B(theta)*R/(c_max + B(theta)*R) - M) # gains
Mort = lambda theta: m0 + mp*theta #np.log(theta+1) # mortality

r = lambda theta: G(theta) - Mort(theta) # growth rate as function of trait theta

max_index = [i for i,v in enumerate(r(theta_list)) if v==max(r(theta_list))][0]
root_r = sp.optimize.root_scalar(r, bracket=[200, 400]).root
```

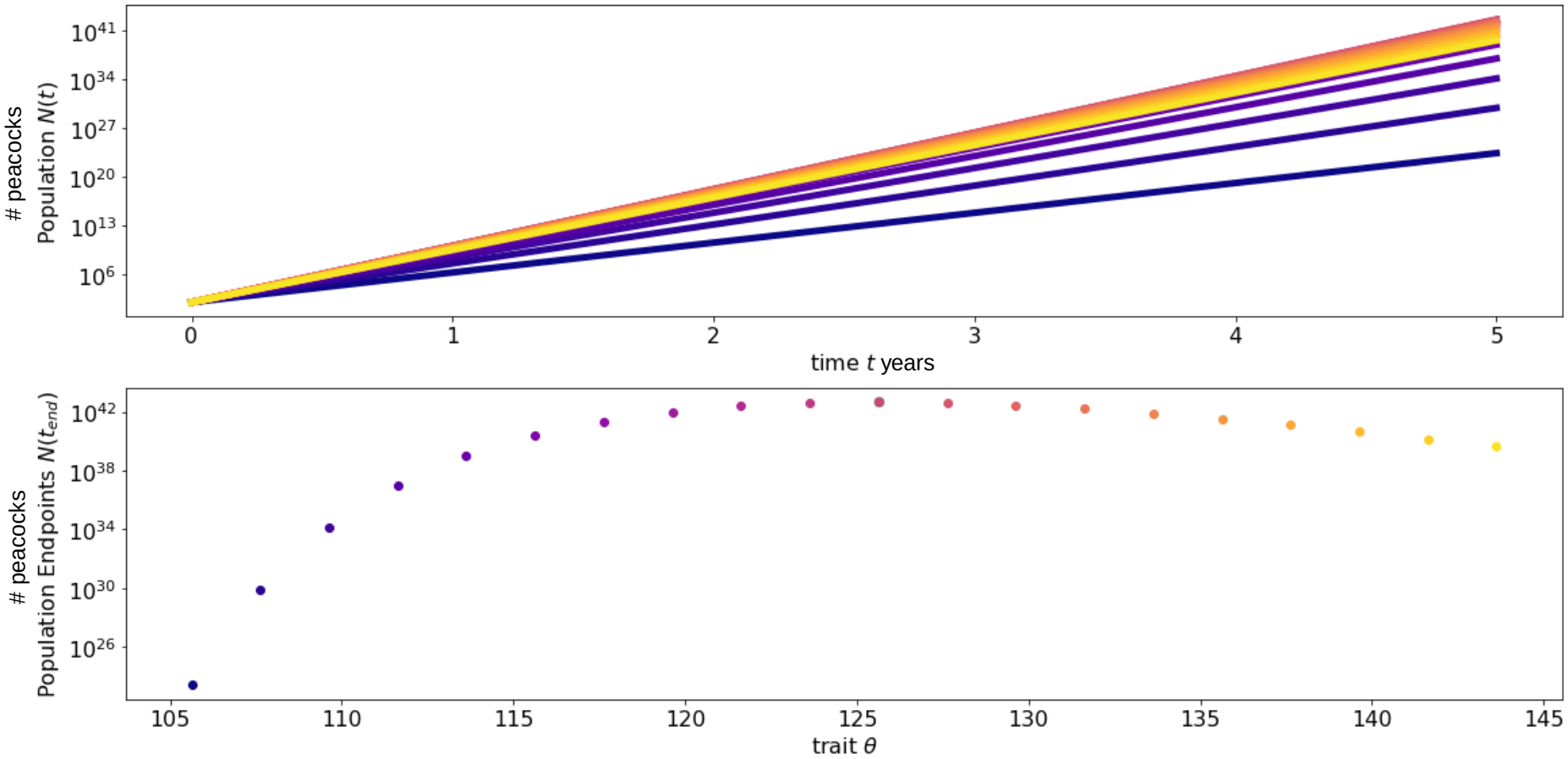
```
fig, ax = plt.subplots(1, figsize=(10,10)) #, sharex=True
ax.plot(theta_list, B(theta_list), color = 'lightgreen', label = 'Mating rate')
ax.axhline(0, linewidth = 2, linestyle = ':', color = 'lightgreen')
ax.set_xlabel("trait $\\theta$")
ax.set_ylabel("Mating rate, units # peahens/ year/ # peacock")
ax.legend()


fig, ax = plt.subplots(3, 1, figsize=(10,30)) #, sharex=True
ax[0].plot(theta_list, G(theta_list), label="Gains", color = 'mediumseagreen')
ax[0].plot(theta_list, Mort(theta_list), label = 'Mortality', color = 'firebrick')
ax[0].axvline(theta_list[max_index], linestyle=":", alpha=0.5, color="magenta", label="optimal growth rate")
ax[0].axvline(root_r, linestyle=":", alpha=0.5, color="cyan", label="zero growth")
ax[0].set_xlabel("trait $\\theta$")
ax[0].set_ylabel("Gains & Mortality, units 1/year")
ax[0].legend(loc="lower right")

# Here we differentiate numerically to find the rate of change
# Basically we find the difference between each two points. So we end up with one point less. So we need one less
theta.
ax[1].plot(theta_list[:-1], np.diff(G(theta_list)), label="$\\frac{d G}{d \\theta}$", color='mediumseagreen')
ax[1].plot(theta_list[:-1], np.diff(Mort(theta_list)), label="$\\frac{d Mort}{d \\theta}$",color='firebrick')
ax[1].axvline(theta_list[max_index], linestyle=":", alpha=0.5, color="magenta", label="optimal growth rate")
ax[1].axvline(root_r, linestyle=":", alpha=0.5, color="cyan", label="zero growth")
ax[1].set_xlabel("trait $\\theta$")
ax[1].set_ylabel("log(rate of change for Gains & Mortality)")
ax[1].legend(loc="lower right")
ax[1].set_yscale("log")


ax[2].plot(theta_list, r(theta_list))
ax[2].axhline (0, linewidth=2, color="k", alpha=0.5)
ax[2].set_xlabel("trait $\\theta$")
ax[2].set_ylabel("growth rate $r(\\theta)$")
ax[2].axvline(theta_list[max_index], linestyle=":", alpha=0.5, color="magenta", label="optimal growth rate")
ax[2].axvline(root_r, linestyle=":", alpha=0.5, color="cyan", label="zero growth")
ax[2].legend(loc="lower right")
```

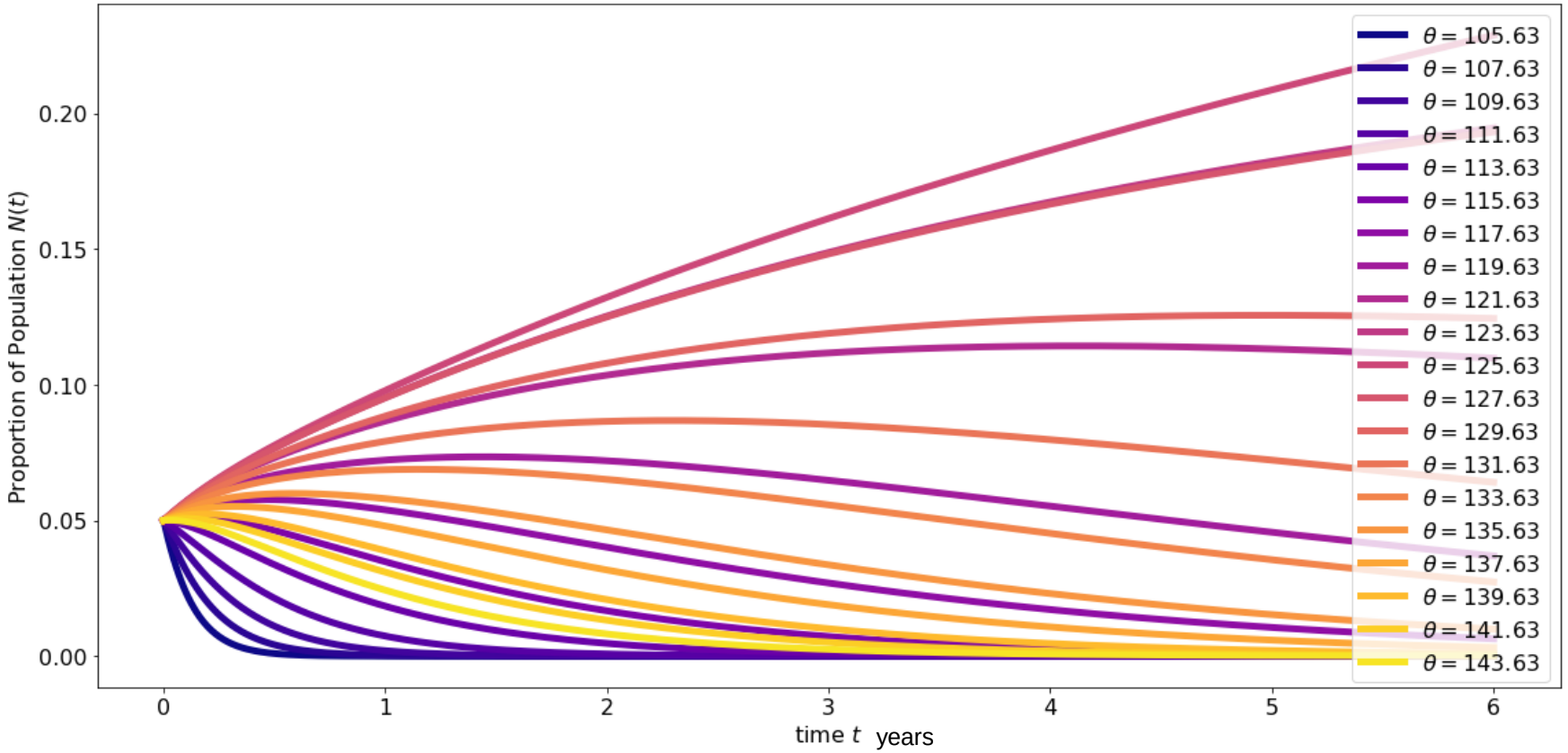
Evolution & optimization: solve the ODE



We solved the ODE for different peacock categories, that differ in the number of eye-spots.

As expected from the analysis beforehand, the peacocks with 125 eye-spots have a higher population at the end point of the simulation.

Evolution & optimization: solve the ODE



Here we are looking at the fractions of each category in the whole peafowl population.

Again, as expected from the analysis beforehand, the peacocks with 125 eye-spots comprise the highest percentage of the population at the end point of the simulation.

Python code

```
# Peacocks numbers

tmp_theta_list = [theta_list[max_index] + i*2 for i in range(-10, 10)] # choose some thetas
state_init = [100]*len(tmp_theta_list) # make the same number of initial populations
params = tuple(tmp_theta_list)

def deriv(state, t, *params):
    return np.array([r(theta)*state[i] for i, theta in enumerate(params)])

t_list = np.linspace(0, 5, 100)
ns = odeint(deriv, state_init, t_list, args=params)



fig, ax = plt.subplots(2, 1, figsize=(16,8), tight_layout=True)
for i, e in enumerate(zip(tmp_theta_list, ns.T)):
    tmp_theta, sol = e
    j = i/len(tmp_theta_list)
    print(tmp_theta)
    if tmp_theta == theta_list[max_index]:
        ax[1].scatter(tmp_theta, sol[-1], s=50, color="green", alpha=0.5, label="optimal growth")
# this dont plot for some reason...
    ax[0].plot(t_list, sol, color=plt.cm.plasma(j), label=f"$\\theta = {round(tmp_theta,2)}$")
    ax[1].scatter(tmp_theta, sol[-1], color=plt.cm.plasma(j), label=f"$\\theta =
{round(tmp_theta,2)}$")

ax[0].set_ylabel("Population $N(t)$")
ax[0].set_xlabel("time $t$")
ax[0].set_yscale("log")

ax[1].set_ylabel("Population Endpoints $N(t_{end})$")
ax[1].set_xlabel("trait $\\theta$")
ax[1].set_yscale("log")
```

```
#Fractions

tmp_theta_list = [theta_list[max_index] + i*2 for i in range(-10, 10)]
state_init = [1/len(tmp_theta_list)]*len(tmp_theta_list)
params = tuple(tmp_theta_list)

def deriv_compare(state, t, *params):
    s = lambda cur_theta: np.sum(np.array([(r(cur_theta) - r(theta))*state[i] for i, theta in
enumerate(params)]))
    pdots = np.array([state[i]*s(theta) for i, theta in enumerate(params)])
    return pdots



# Long time scale:

t_list = np.linspace(0, 6, 1000)
ns = odeint(deriv_compare, state_init, t_list, args=params)

fig, ax = plt.subplots(figsize=(16,8), tight_layout=True)
for i, e in enumerate(zip(tmp_theta_list, ns.T)):
    tmp_theta, sol = e
    j = i/len(tmp_theta_list)
    ax.plot(t_list, sol, color=plt.cm.plasma(j), label=f"$\\theta = {round(tmp_theta,2)}$")

ax.set_ylabel("Proportion of Population $N(t)$")
ax.set_xlabel("time $t$")
ax.legend()
```