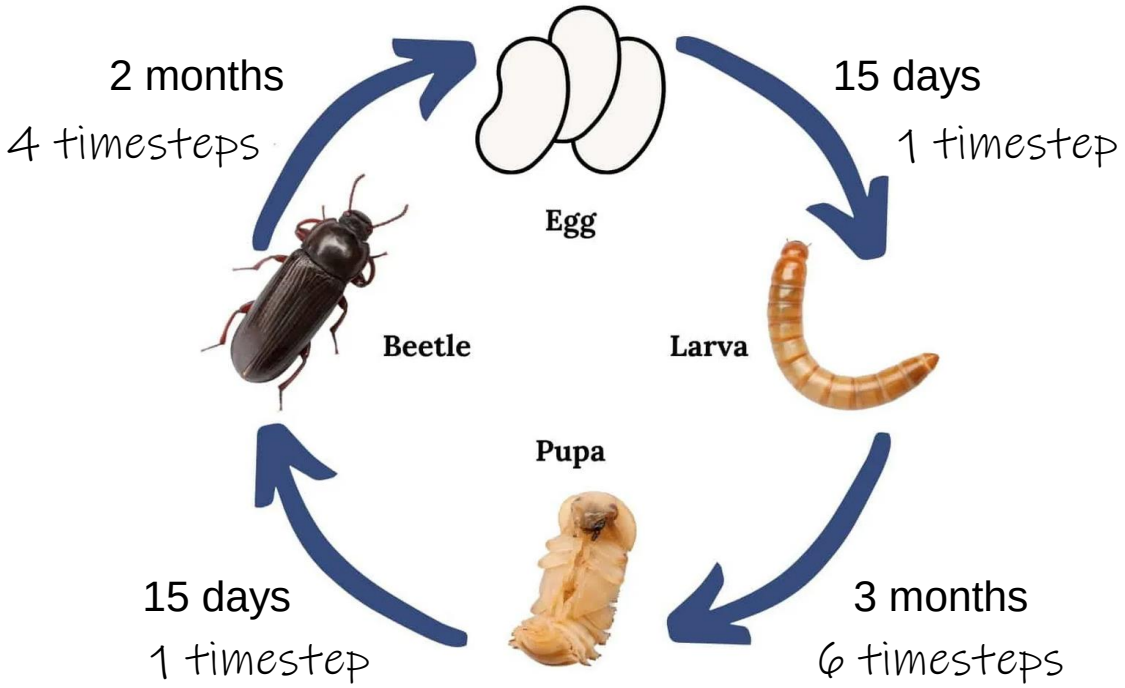
Group: Amalia Bogri, Christian Berrig & Jonas Bolduan

Structured population models

Stage-structured population: *Tenebrio molitor*



Beetles are holometabolous insects,
they go through metamorphosis

The four life stages are:
egg, larva, pupa and adult beetle

We will use a stage-structured model

Our time step is 15 days (half a month)

Only the adults are sexually mature

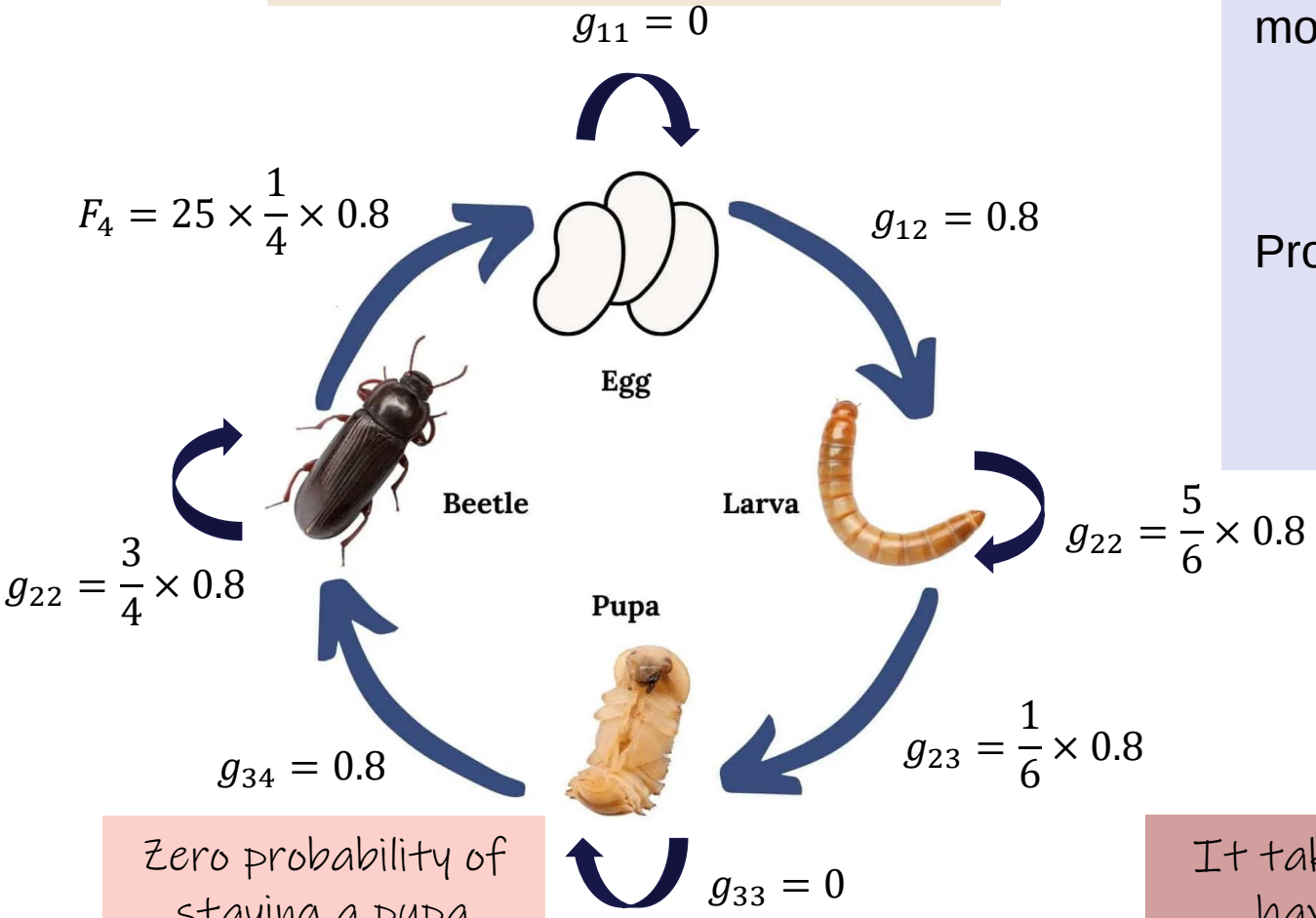
We assume that each adult produces 25 eggs

Dimension of each life stage: number of individuals

Image source: https://www.heritageacresmarket.com/mealworm-farm/

Stage-structured population: *Tenebrio molitor*

Zero probability of staying an egg (die or become larva)



Zero probability of staying a pupa

To keep a 'realistic' population growth we introduce mortality:

20% of the individuals of each life-stage die (i.e. $d = 0.2$ probability of death)

Probability of survival of each stage is $s = 0.8$,
and is comprised of
probability of staying in the same stage g_{xx}
& probability of moving to next stage $g_{x\,x+1}$

The transition from larva to pupa takes 6 timesteps.
We translate this into a 1/6 chance of the larvae to transform in one time step,
and 5/6 to remain larvae.

It takes 4 timesteps for adults to produce eggs, thus they have ¼ chance of producing 25 eggs in each time step

Image source: https://www.heritageacresmarket.com/mealworm-farm/

Stage-structured population: *Tenebrio molitor*

4 stage structures:

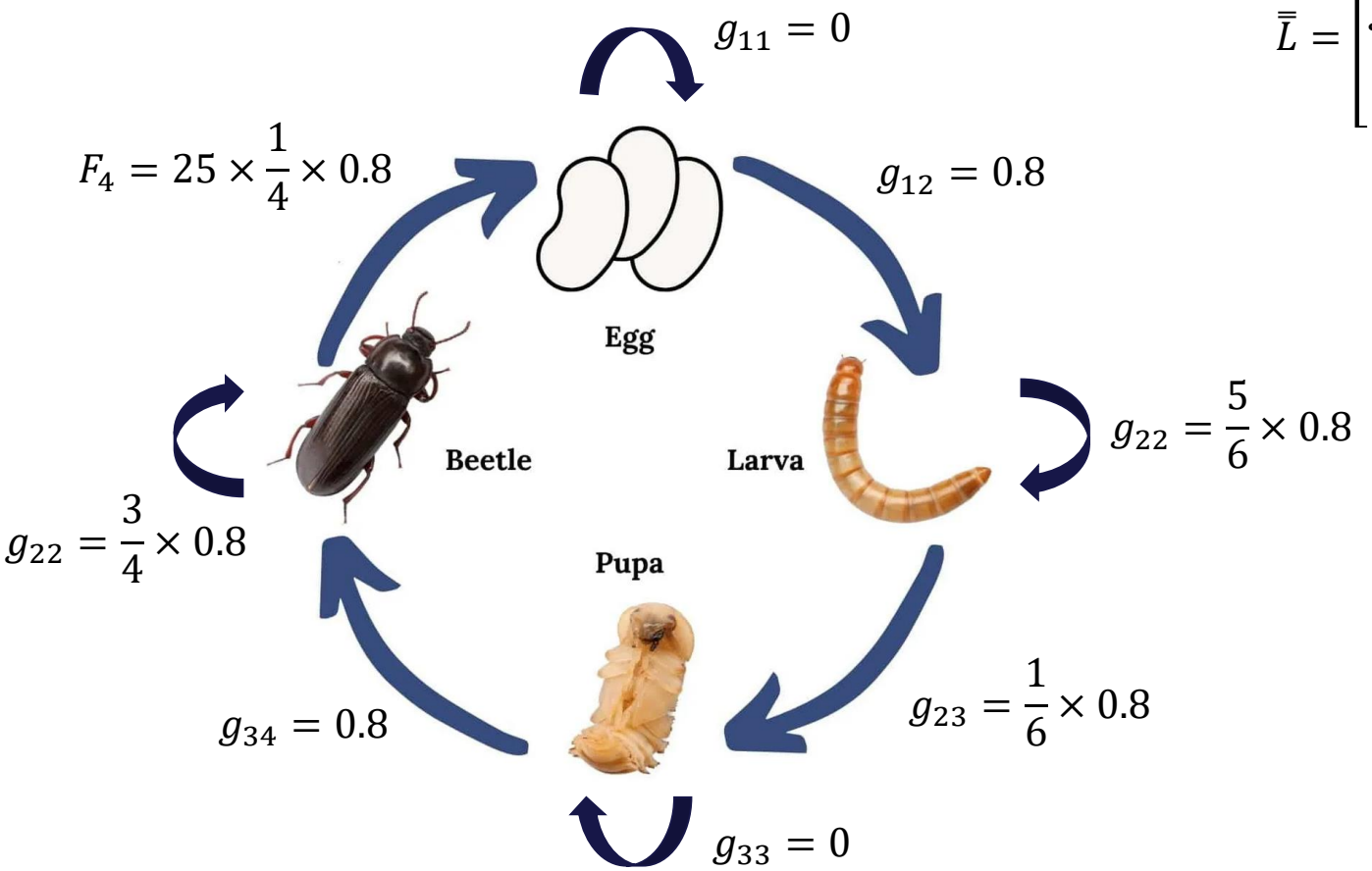


Image source: https://www.heritageacresmarket.com/mealworm-farm/

Matrix with model parameters:

$$\bar{\bar{L}} = \begin{bmatrix} g_{11} & 0 & 0 & F_4 \\ g_{12} & g_{22} & 0 & 0 \\ 0 & g_{23} & g_{33} & 0 \\ 0 & 0 & g_{34} & g_{44} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 25 \cdot 0.8 \cdot 1/4 \\ 0.8 & 0.8 \cdot 5/6 & 0 & 0 \\ 0 & 0.8 \cdot 1/6 & 0 & 0 \\ 0 & 0 & 0.8 & 0.8 \cdot 3/4 \end{bmatrix}$$

Initial conditions: $\bar{n}_{t=0} = \begin{bmatrix} 100 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

The eigenvalues of the matrix $\bar{\bar{L}}$ are complex numbers, so we expect **oscillatory behaviour**.

-0.55151171+0i

0.31639741+0.74332683i

0.31639741-0.74332683i

1.18538356+0i

The largest eigenvalue is >1, so we expect the **population to grow** in time.

Stage-structured population: *Tenebrio molitor*

We calculate the number of individuals at each stage by iterating for 40 steps (i.e. 20 months) through:

$$\begin{bmatrix} n_{egg} \\ n_{larva} \\ n_{pupa} \\ n_{adult} \end{bmatrix}(t) = \begin{bmatrix} g_{11} & 0 & 0 & F_4 \\ g_{12} & g_{22} & 0 & 0 \\ 0 & g_{23} & g_{33} & 0 \\ 0 & 0 & g_{34} & g_{44} \end{bmatrix} \begin{bmatrix} n_{egg} \\ n_{larva} \\ n_{pupa} \\ n_{adult} \end{bmatrix}(t-1)$$

We find total population N by summing the individuals of each life stage. ★

We calculate the exponential population growth in continuous time with the dominant eigenvalue λ_{dom}:

$$\frac{dN}{dt} = N_0 e^{rt}$$

where growth rate: $r = \ln \lambda_{dom}$

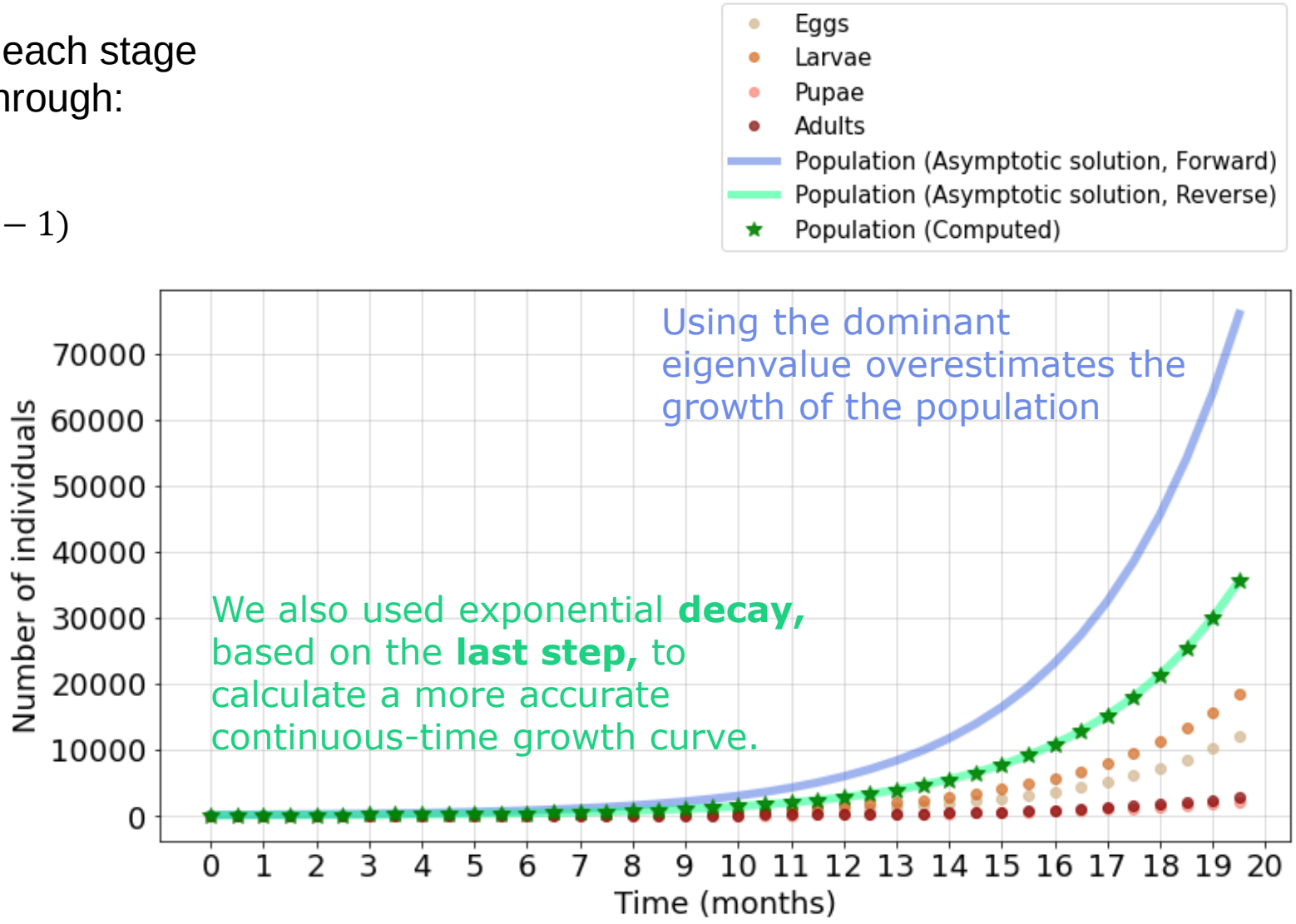
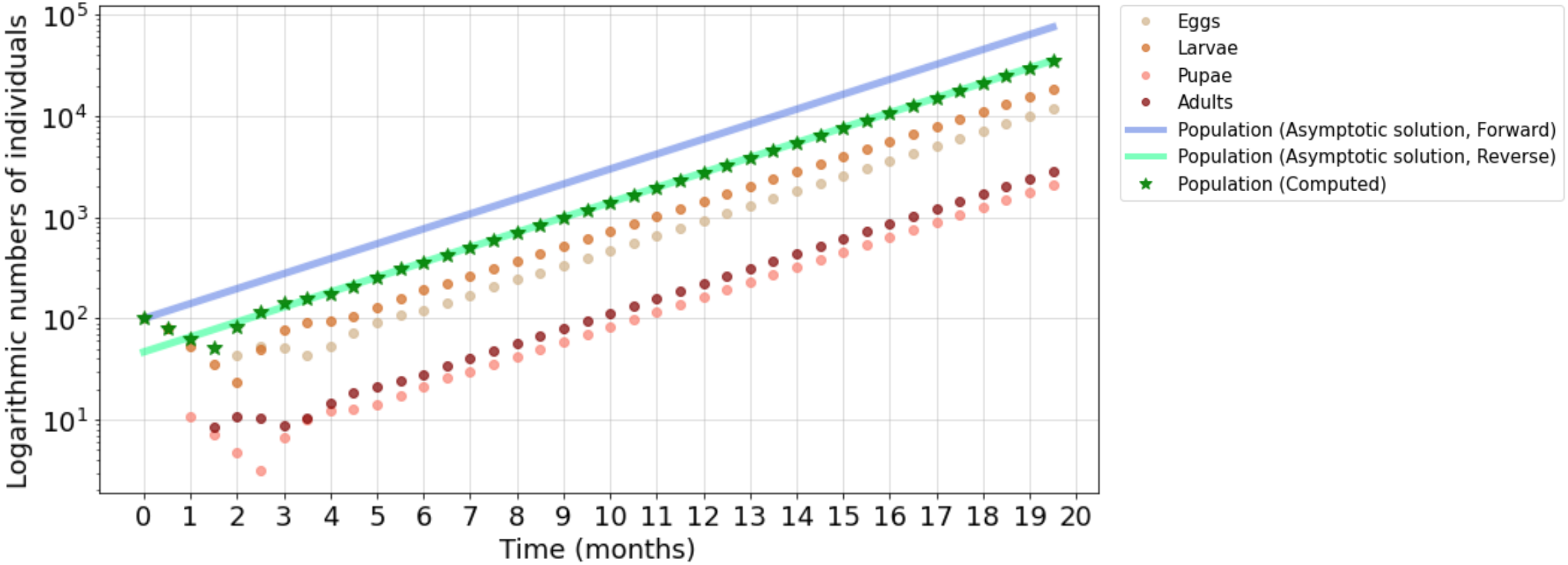


Image source: https://www.heritageacresmarket.com/mealworm-farm/

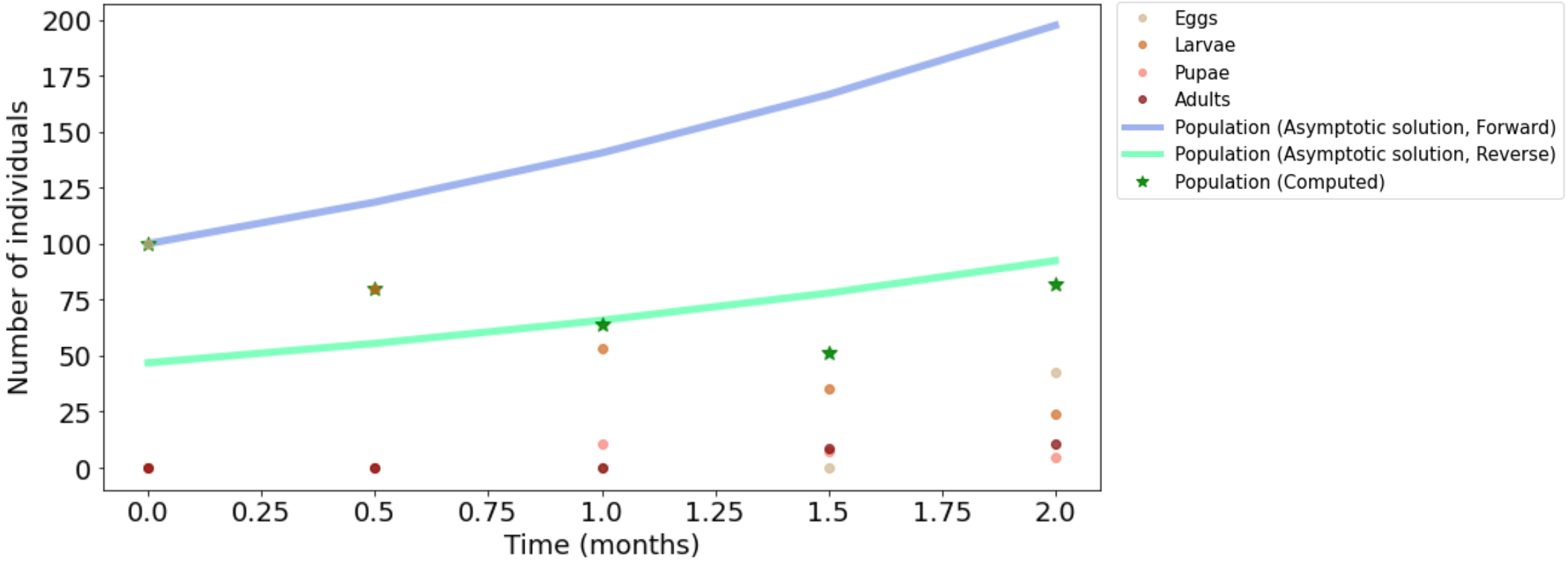
Stage-structured population: *Tenebrio molitor*

Using **semi-logarithmic axes,** we can see that after the stable stage-structure is reached, the growth of each life stage on the ln-scale becomes linear and the lines become parallel.



Stage-structured population: *Tenebrio molitor*

Zooming in the first 2 months of the simulation, we can observe the oscillation(s) of the system.

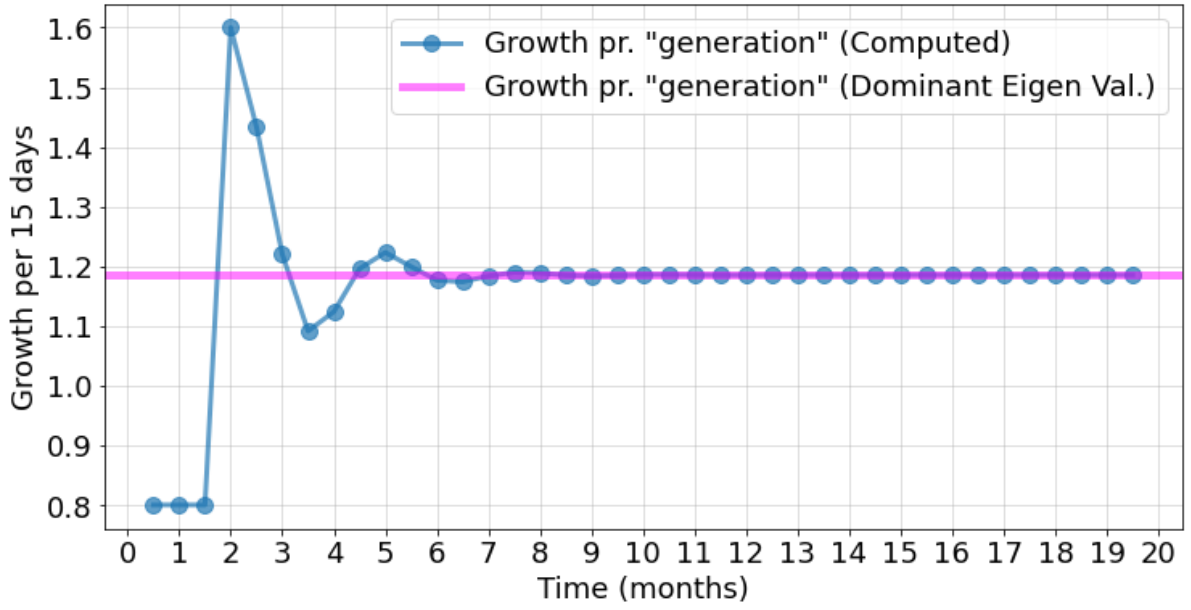


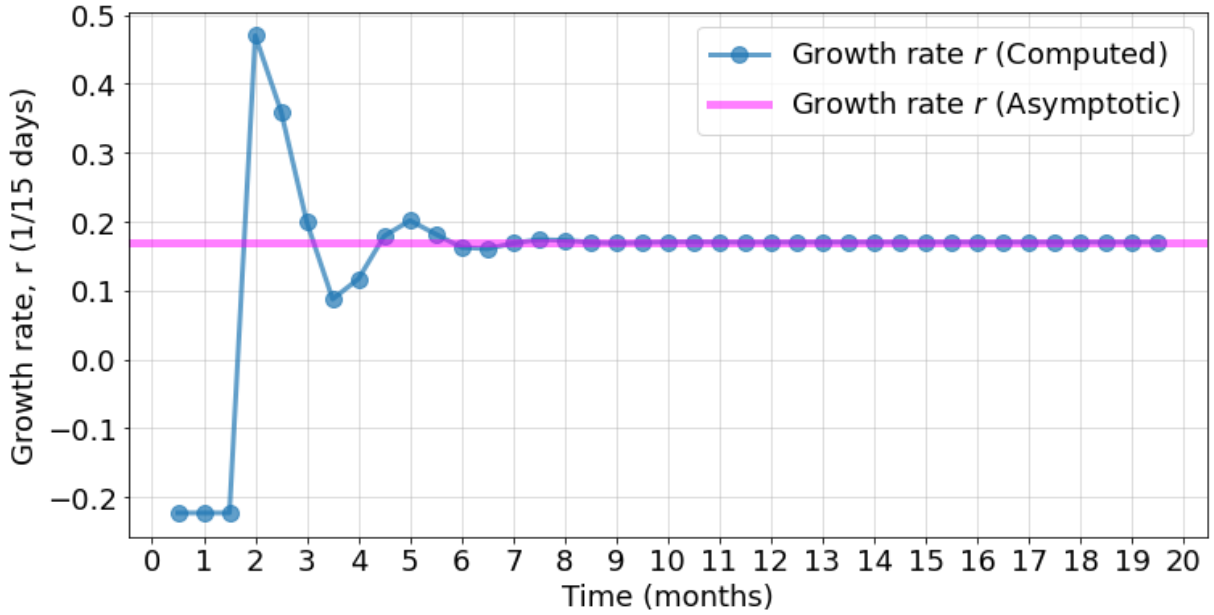
Stage-structured population: *Tenebrio molitor*

We calculate the growth rate per generation in the discrete time as: n_{t-1}/n_t
In continuous time as: $\lambda_{con} = \ln(n_{t-1}/n_t)$

We compare it to the growth rate predicted by the dominant eigenvalue $\lambda_{dom} = 1.18$ and $r = ln\,\lambda_{dom}$

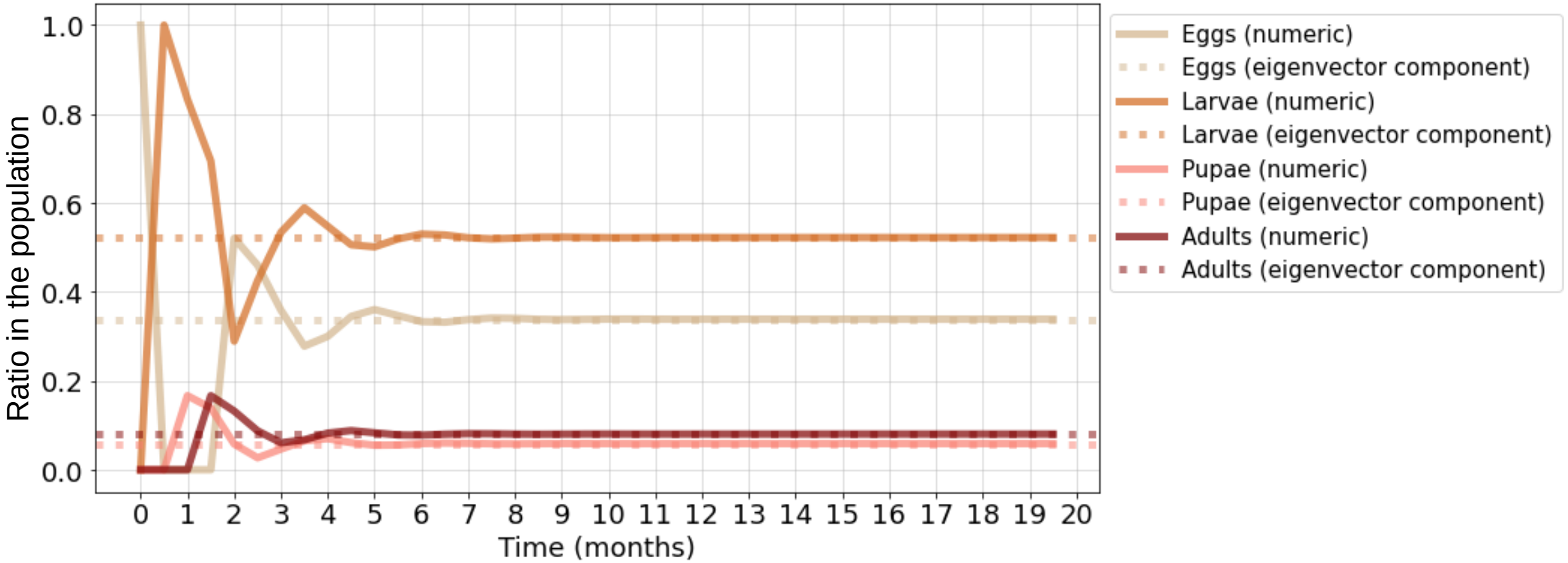
As expected, in both cases, the computed growth rates eventually reach an equilibrium at the value predicted by the dominant eigenvalue.





Stage-structured population: *Tenebrio molitor*

Plotting the **ratio of each life stage** in the population, it is clear that the **stable stage-structure** is reached. The stable ratios for each stage coincide with the ratio of the eigenvector of the dominant eigenvalue to the sum of these eigenvectors, for each life stage.



Stage-structured population: Density dependence

We attempt to introduce a density dependence in the fecundity term.
Now fecundity is defined as by:

$$F(N) = 25\left(\frac{1}{1+aN}\right)$$

where $a = 1$

$$\bar{\bar{L}} = \begin{bmatrix} 0 & 0 & 0 & \boldsymbol{F(N)} \cdot 0.8 \cdot 1/4 \\ 0.8 & 0.8 \cdot 5/6 & 0 & 0 \\ 0 & 0.8 \cdot 1/6 & 0 & 0 \\ 0 & 0 & 0.8 & 0.8 \cdot 3/4 \end{bmatrix}$$

With the current choice of parameters, the **population declines** in time.

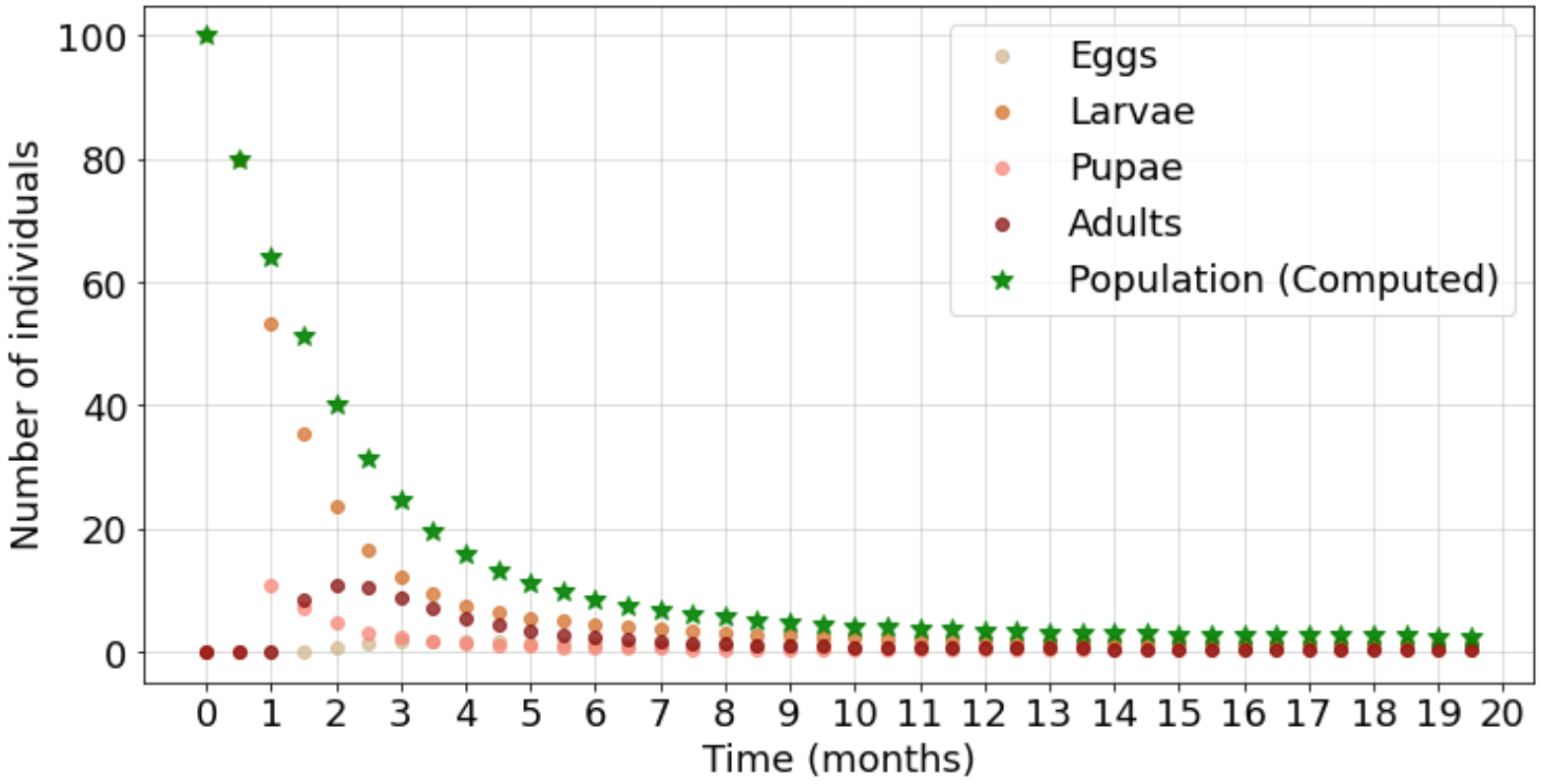
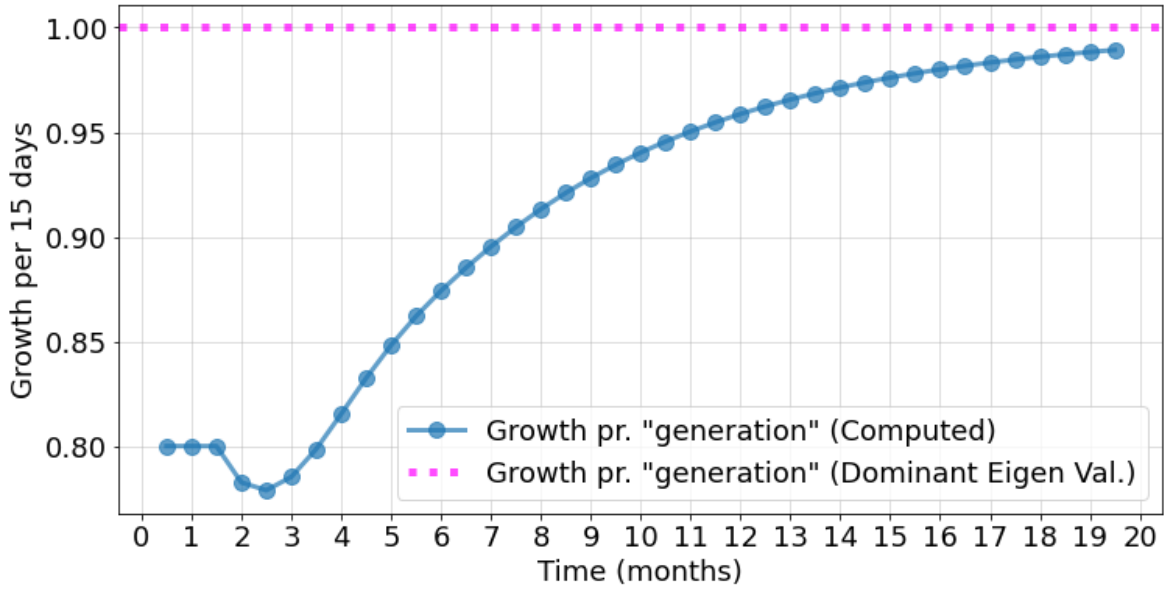


Image source: https://www.heritageacresmarket.com/mealworm-farm/

Stage-structured population: Density dependence

For this system, the equilibrium condition is achieved at a dominant eigenvalue $\boldsymbol{\lambda_{dom} = 1}$ (since the system stops growing and r should be zero).

Examining the growth rates in discrete and continuous time, we see, again, that they eventually approach the dominant eigenvalue and its respective r.



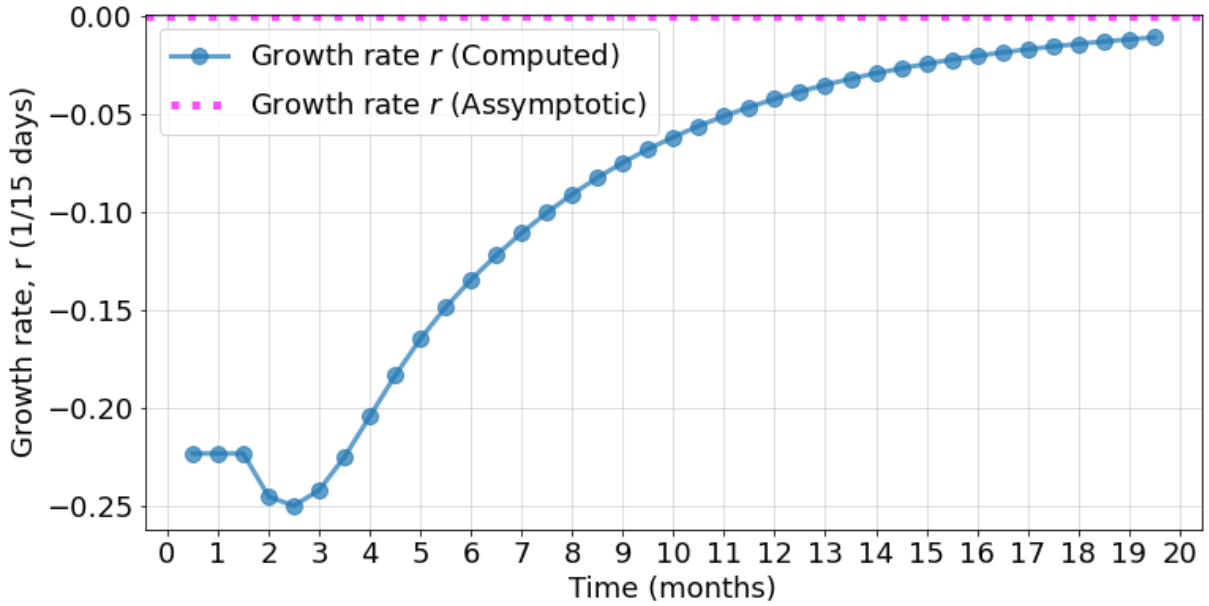


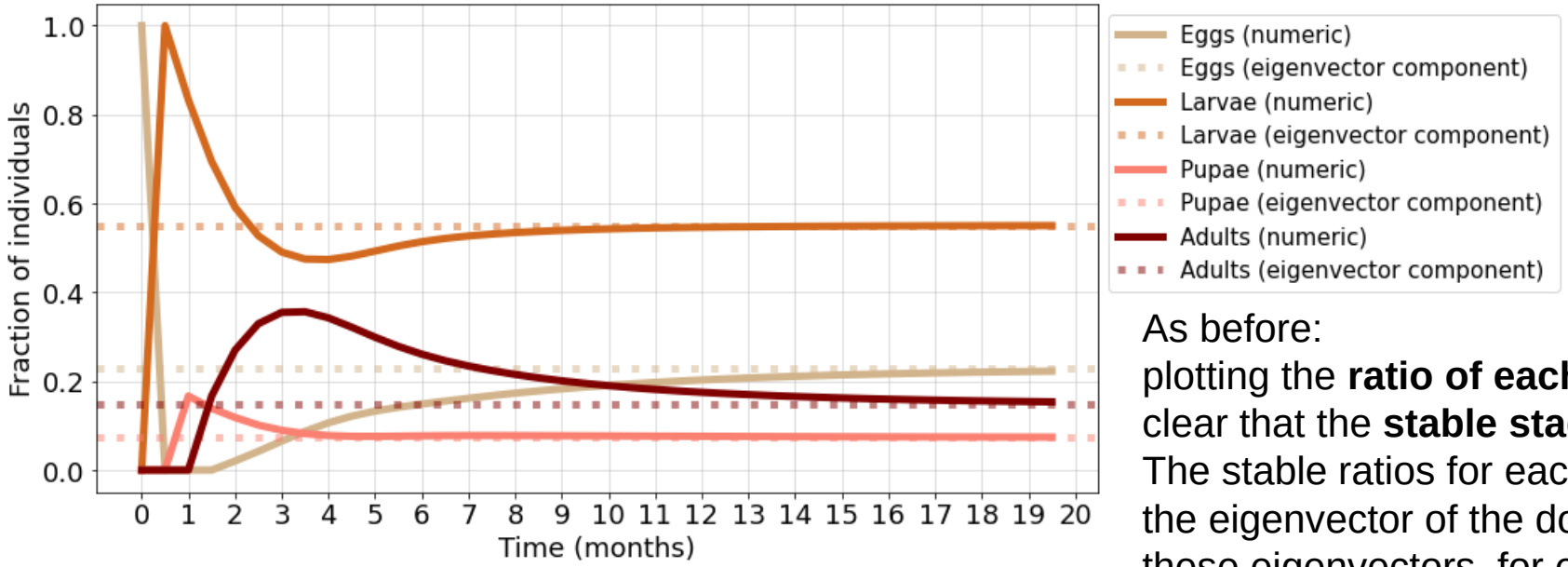
Image source: https://www.heritageacresmarket.com/mealworm-farm/

Stage-structured population: Density dependence

By calculating the value of $F(N)$ for the equilibrium, F^*, we can calculate the eigenvectors of the dominant eigenvalue.

$F^* = (5 \cdot 0.8\text{-}6) \cdot (3 \cdot 0.8\text{-}4)/(0.8^4)$

$$\bar{\bar{L}} = \begin{bmatrix} 0 & 0 & 0 & \boldsymbol{F^*} \cdot 0.8 \cdot 1/4 \\ 0.8 & 0.8 \cdot 5/6 & 0 & 0 \\ 0 & 0.8 \cdot 1/6 & 0 & 0 \\ 0 & 0 & 0.8 & 0.8 \cdot 3/4 \end{bmatrix}$$



As before:
plotting the **ratio of each life stage** in the population, it is clear that the **stable stage-structure** is reached.
The stable ratios for each stage coincide with the ratio of the eigenvector of the dominant eigenvalue to the sum of these eigenvectors, for each life stage.

Image source: https://www.heritageacresmarket.com/mealworm-farm/

Stage-structured population: Density dependence

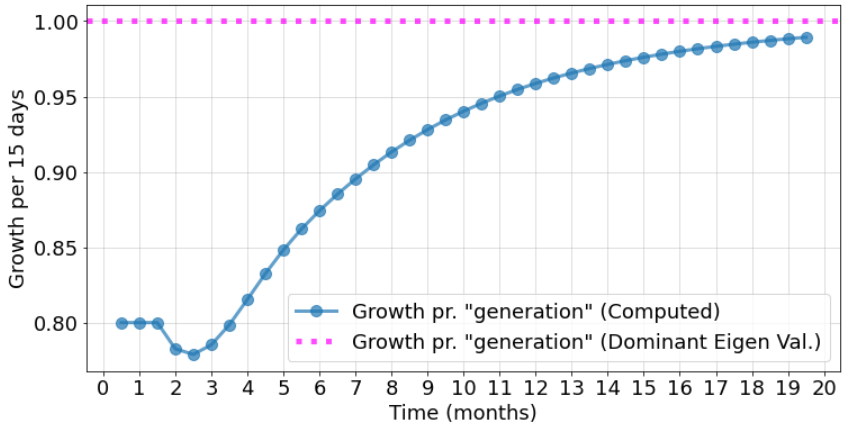
Notably, the value of F^* is the what matters for the values of the system in the stable stage structure, and not the actual form of $F(N)$. Of course, though, the transient stage is dependent on $F(N)$.

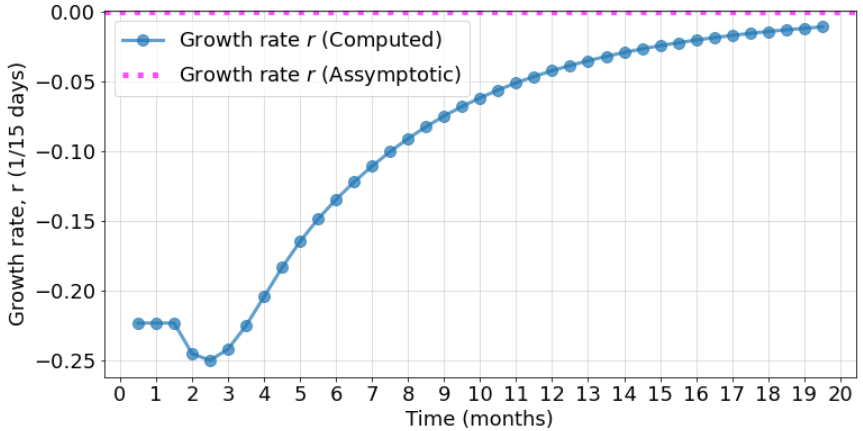
We compare the previous results with the ones using the new equation $F(N) = 25e^{-aN}$, where $a = 1$

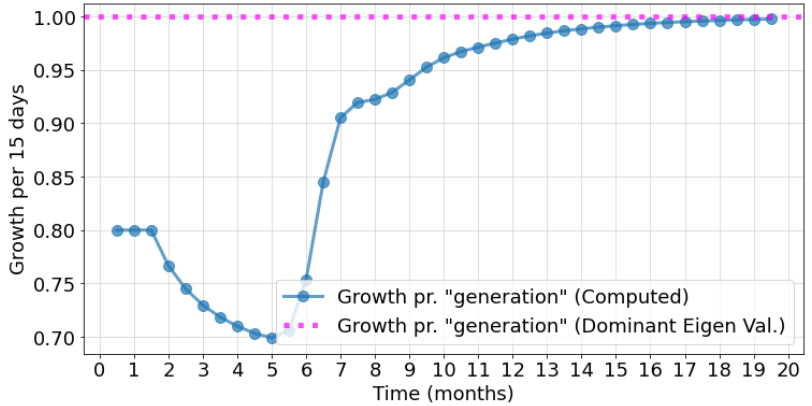
$$F(N) = 25\left(\frac{1}{1+aN}\right)$$

$$F(N) = 25e^{-aN}$$

Transient phases different, stable phases very similar.







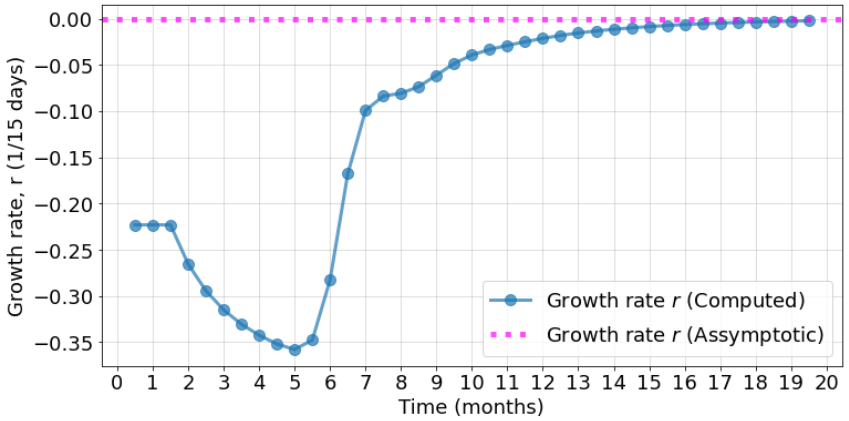
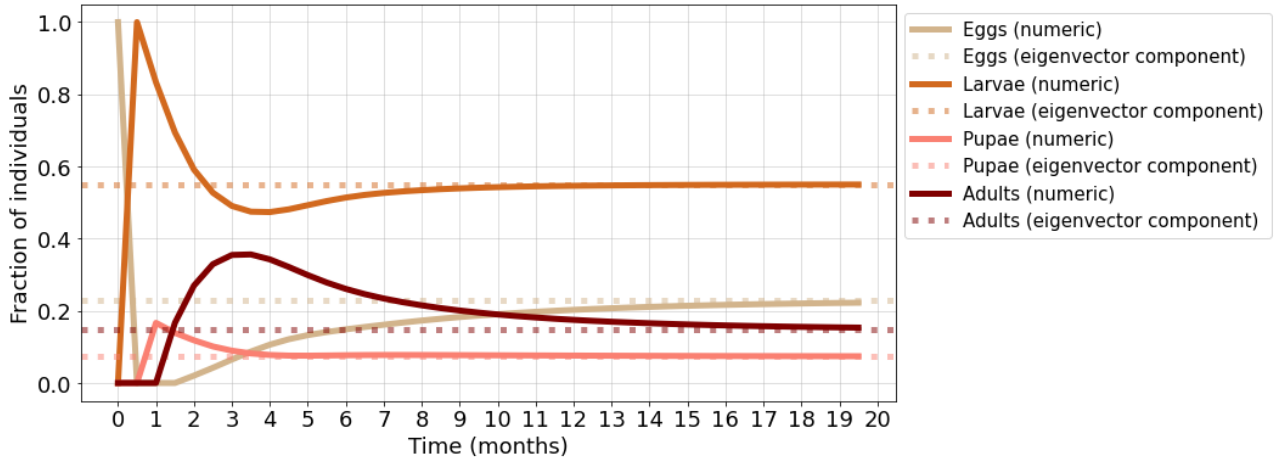


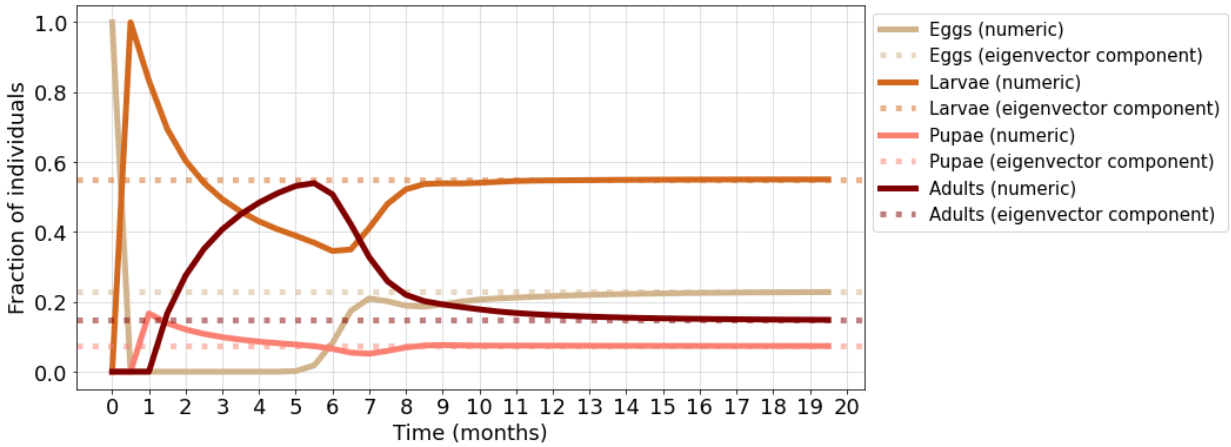
Image source: https://www.heritageacresmarket.com/mealworm-farm/

Stage-structured population: Density dependence

$$F(N) = 25\left(\frac{1}{1+aN}\right)$$



$$F(N) = 25e^{-aN}$$



Transient phases different, stable phases very similar.

Image source: https://www.heritageacresmarket.com/mealworm-farm/

Code in Python

```
from scipy.integrate import odeint
import matplotlib.pyplot as plt
import numpy as np

# This is for reasonable fontsize universally defined:
fs_label = 18
parameters = {
                'figure.titlesize': fs_label+6,
                'axes.labelsize': fs_label,
                'axes.titlesize': fs_label+6,
                'xtick.labelsize': fs_label,
                'ytick.labelsize': fs_label,
                'legend.fontsize': fs_label,
                'lines.linewidth': 5,
}
plt.rcParams.update(parameters)

L = np.array([
    [0,   0,       0,   25*0.8*1/4],
    [0.8, 0.8*5/6, 0,   0],
    [0,   0.8/6,   0,   0],
    [0,   0,       0.8, 0.8*3/4]
])
# getting eigen vals + vects
eig_res = np.linalg.eig(L)
lams, vecs = eig_res # extract the arrays into variables

# getting only the dominatingeigen vals + corresponding eigrn vect.
id_max = max(enumerate([abs(x) for x in lams]), key=lambda x: x[1])[0]
#gives index of max eigenvalue
lam_dom, vec_dom = lams[id_max], vecs[:, id_max]

#defining max growth-rate
r = np.log(lam_dom)
```

```
def time_series(n0, t_end=10):
    ret = np.zeros((t_end, len(lams)))
    n = n0
    ret[0] = n0
    for t in range(1, t_end):
        n = np.matmul(L, n)
        ret[t] = n
    return ret

# Fix time:
t_end = 40 # how many steps
t = np.arange(t_end) # make array with 40 steps
xx_ticks = np.linspace(0, 40, 21)
xx_labels = [int(i/2) for i in xx_ticks] # for plot: make the labels for the x axis ticks

state_init = np.array([100, 0, 0, 0]) # set initial number of individuals for the 4 stages
sim_res = time_series(state_init, t_end=t_end) # solve the function
pop = np.sum(sim_res, axis=1) # find total population:
growth = pop[1:]/pop[:-1] # find growth rate:
proportions = np.array([s/np.sum(s) for s in sim_res]) # find the proportions of each stage

discrete_style = {"linestyle":None, "linewidth":0, "marker":'o', "markersize":6, "alpha":0.7}
labels = ["Eggs", "Larvae", "Pupae", "Adults"]
colors = ['tan', 'chocolate', 'salmon', 'maroon']

fig, ax = plt.subplots(figsize=(12, 6))
for i, state in enumerate(sim_res.T):
    ax.plot(t, state, label=labels[i], color=colors[i], **discrete_style)
ax.plot(t, pop[0]*np.exp(r*t), label="Population (Asymptotic solution, Forward)",
color="royalblue", alpha=0.5)
ax.plot(t, pop[-1]*np.exp(-r*t)[::-1], color='springgreen', label="Population (Asymptotic
solution, Reverse)", alpha=0.5)
ax.plot(t, pop, label="Population (Computed)", color='green', marker = '*', markersize =
10,alpha = 0.9, linewidth = 0)
ax.set_xticks(xx_ticks)
ax.set_xticklabels(xx_labels)
ax.legend(bbox_to_anchor=(1.0, 1), loc=2, borderaxespad=0.5, fontsize=15)
ax.set_xlabel("Time (months)")
ax.set_ylabel("Number of individuals")
ax.grid(alpha = 0.5)
ax.set_yscale('log') # for second graph only!
```

Code in Python

```
# Zoom in the plot
end_tmp = 5
xx_ticks2 = np.linspace(0, 4, 9)
xx_labels2 = [i/2 for i in xx_ticks2]

fig, ax = plt.subplots(figsize=(12, 6))

ax.plot(t[:end_tmp], pop[0]*np.exp(r*t)[:end_tmp], label="Population
(Assymptotic solution, Forward)", color="royalblue", alpha=0.5)
ax.plot(t[:end_tmp], pop[-1]*np.exp(-r*t)[::-1][:end_tmp], label="Population
(Assymptotic solution)", color='springgreen', alpha=0.5)
ax.plot(t[:end_tmp], pop[:end_tmp], label="Population
(Computed)",color='green', marker = '*', markersize = 10,alpha = 0.9, linewidth
= 0)

for i, state in enumerate(sim_res.T):
    ax.plot(t[:end_tmp], state[:end_tmp], label=labels[i], color=colors[i],
**discrete_style)

ax.set_xticks(xx_ticks2)
ax.set_xticklabels(xx_labels2)
ax.set_xlabel("Time (months)")
ax.set_ylabel("Number of individuals")
ax.legend(bbox_to_anchor=(1.0, 1), loc=2, borderaxespad=0.5, fontsize=15)
```

```
fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(t[1:], growth, label="Growth pr. \"generation\" (Computed)", alpha=0.7,
marker=".", markersize=20, linewidth=3)
ax.axhline(lam_dom, label="Growth pr. \"generation\" (Dominant Eigen Val.)",
alpha=0.5, color="magenta")
ax.legend()
ax.set_xticks(xx_ticks)
ax.set_xticklabels(xx_labels)
ax.set_xlabel("Time (months)")
ax.set_ylabel("Growth per 15 days")
ax.grid(alpha = 0.5)
```

```
fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(t[1:], np.log(growth), label="Growth rate $r$ (Computed)", alpha=0.7,
marker=".", markersize=20, linewidth=3)
ax.axhline(r, label="Growth rate $r$ (Asymptotic)", alpha=0.5, color="magenta")
ax.legend()
ax.set_xticks(xx_ticks)
ax.set_xticklabels(xx_labels)
ax.set_xlabel("Time (months)")
ax.set_ylabel("Growth rate, r (1/15 days)")
ax.grid(alpha = 0.5)
```

```
fig, ax = plt.subplots(figsize=(12, 6))
for i, v in enumerate(vec_dom):
    num_prop = ax.plot(t, np.array(proportions).T[i], color=colors[i],
label=labels[i]+" (numeric)", alpha = 0.7)
    ax.axhline(v/sum(vec_dom), alpha=0.5, linestyle=":",
color=num_prop[0].get_color(), label=labels[i]+" (eigenvector component)")
ax.set_xticks(xx_ticks)
ax.set_xticklabels(xx_labels)
ax.legend(bbox_to_anchor=(1.0, 1), loc=2, borderaxespad=0.5, fontsize=15)
ax.set_xlabel("Time (months)")
ax.set_ylabel("Percentage in the population")
ax.grid(alpha = 0.5)

# disregard warning, imaginary part is zero...
```

Code in Python

```
# Question 3: Density dependence

#This defines the fecundity:
a = 1
# f = lambda N: 25*np.exp(-a*N)
f = lambda N: 25*(1/(1 + a*N))
#...and use this functional dependency in the "next-gen" matrix.
L = lambda N: np.array([
    [0,   0,       0,   f(N)*0.8*1/4],
    [0.8, 0.8*5/6, 0,   0],
    [0,   0.8*1/6, 0,   0],
    [0,   0,       0.8, 0.8*3/4]
])


def time_series_density(n0, t_end=10):
    ret = np.zeros((t_end, len(lams)))
    n = n0
    ret[0] = n0
    for t in range(1, t_end):
        N = np.sum(n)
        n = np.matmul(L(N), n)
        ret[t] = n
    return ret

state_init = np.array([100, 0, 0, 0])
sim_res = time_series_density(state_init, t_end=t_end)
pop = np.sum(sim_res, axis=1)
growth = pop[1:]/pop[:-1]
proportions = np.array([s/np.sum(s) for s in sim_res])
```

```
fig, ax = plt.subplots(figsize=(12, 6))
for i, state in enumerate(sim_res.T):
    ax.plot(t, state, label=labels[i],color=colors[i], **discrete_style)
ax.plot(t, pop, label="Population (Computed)", color='green', marker = '*', markersize
= 10,alpha = 0.9, linewidth = 0)

ax.legend()
ax.set_xticks(xx_ticks)
ax.set_xticklabels(xx_labels)
ax.set_xlabel("Time (months)")
ax.set_ylabel("Number of individuals")
ax.grid(alpha = 0.5)

# found the fecundity for eigenvalue=1, that is equilibrium condition.
f_star = (5*0.8-6)*(3*0.8-4)/(0.8**4)
L_star = np.array([
    [0,   0,       0,   f_star*0.8*1/4],
    [0.8, 0.8*5/6, 0,   0],
    [0,   0.8*1/6, 0,   0],
    [0,   0,       0.8, 0.8*3/4]
])

e, v = np.linalg.eig(L_star)
e_dom, v_dom = e[3], v[:,3]

# THEN WE DID THE SAME PLOTS AS BEFORE (IN THE PROVIDED CODE), SO I WILL NOT WRITE IT
AGAIN HERE AS IT IS LONG
# AFTERWARDS WE DID THE SAME THING WITH THE SECOND F EQUATION AND PLOTTED WITH THE
EXACT SAME WAY
```