

Introduction to Computational Physics – Exercise 7

Simon Groß-Bölting, Lorenz Vogel, Sebastian Willenberg

June 12, 2020

Population dynamics

The simple equation of growth of a population, as proposed by Malthus (1798), has been improved by Verhulst (1836) including a growth limiting term, which represents the finite amount of resources available. In this exercise we study a modified form of Verhulst's equation for population dynamics:

$$\frac{dN}{dt} = rN(1 - N/K) - \frac{BN^2}{A^2 + N^2} \quad (1)$$

where all parameters r , K , A and B are positive. It is a more complex example, in which the growth behaviour depends on whether N is smaller or larger than a critical populations size A .

Dimensional analysis

Task: Determine the dimension of the parameters and rewrite the equation in dimensionless form. Note that there are different possibilities. Please formulate a dimensionless time τ that is not defined on the basis of r . Use $n = N/A$ as the dimensionless version of N .

Table 1: Dimension of the parameters

description	parameter	dimension
time	t	$[t] = \text{s}$
population number	N	$[N] = 1$
growth rate	r	$[r] = \text{s}^{-1}$
growth limiting number	K	$[K] = 1$
critical population size	A	$[A] = 1$
—	B	$[B] = \text{s}^{-1}$

By dimensional analysis we can renormalize the variables, such that the above equation for population dynamics becomes dimensionless: The steps required for this are well described in the Wikipedia article “Nondimensionalization”. First of all we have to identify all the independent and dependent variables in the given equation:

$$\frac{dN}{dt} = rN \left(1 - \frac{N}{K}\right) - \frac{BN^2}{A^2 + N^2} = rN - \frac{rN^2}{K} - \frac{BN^2}{A^2 + N^2} \quad (2)$$

In this equation the independent variable is t (time) and the dependent variable is $N = N(t)$ (population number). Now we replace each of the two variables with the product of a dimensionless variable (τ and n) and a characteristic unit (N_c and t_c):

$$N := N_c n \quad \Leftrightarrow \quad n := N/N_c \quad \text{and} \quad t := t_c \tau \quad \Leftrightarrow \quad \tau := t/t_c \quad (3)$$

The subscripted “c” added to a variable-name is used to denote the characteristic unit used to scale that variable. As required in the task, we set $N_c := A$, so that we use $n := N/A$ as the dimensionless version of the population number N . Then we obtain the following equation:

$$\frac{A}{t_c} \frac{dn}{d\tau} = rAn - \frac{rA^2}{K} n^2 - \frac{BA^2n^2}{A^2 + A^2n^2} = rAn - \frac{rA^2}{K} n^2 - B \frac{n^2}{1 + n^2} \quad (4)$$

Dividing the entire equation by the coefficient A/t_c in front of the first derivative term and using the relation

$$\frac{n^2}{1+n^2} = \left[\frac{1+n^2}{n^2} \right]^{-1} = \left[1 + \frac{1}{n^2} \right]^{-1} = [1+n^{-2}]^{-1} = \frac{1}{1+n^{-2}} \quad (5)$$

gives us:

$$\frac{dn}{d\tau} = t_c r n - \frac{t_c r A}{K} n^2 - \frac{t_c B}{A} \frac{1}{1+n^{-2}} \quad (6)$$

Now the characteristic unit for each variable can be defined, such that the coefficients of as many terms as possible become 1. Since we have already determined the dimensionless version of N , we only have to define the characteristic unit t_c of the time t . Because we are supposed to formulate a dimensionless time $\tau = t/t_c$ that is not defined on the basis of the growth rate r , we have to use the coefficient in front of the last term:

$$\frac{t_c B}{A} \stackrel{!}{=} 1 \quad \implies \quad t_c := \frac{A}{B} \quad (7)$$

Now we can rewrite the given equation for population dynamics in its dimensionless form:

$$\frac{dn}{d\tau} = \frac{rA}{B} n - \frac{rA^2}{BK} n^2 - \frac{1}{1+n^{-2}} \quad (8)$$

By defining the dimensionless parameter $D := rA/B$, we can further simplify the dimensionless equation:

$$\frac{dn}{d\tau} = Dn - \frac{DA}{K} n^2 - \frac{1}{1+n^{-2}} \quad \text{with} \quad D := \frac{rA}{B} \quad (9)$$

Because r , A and B should be positive, the dimensionless parameter D must also be positive.

Stationary points

Task: Determine the stationary points n^* for $K/A = 7.3$. Note that for $n^* \neq 0$ these values are solutions of a cubic equation; it depends on n and the remaining free parameter. The cubic equation should be derived by yourself analytically; its zero points you can obtain numerically / graphically by using e.g. *Mathematica*. When do one or three real solutions exist as a function of the remaining free parameter? (Hint: we do not ask for some analytical formula here! It is enough to vary the free parameter and check using *Mathematica* which three solutions for the stationary points you get; as stationary points only real solutions are valid. Only one digit after the comma is enough, in other words you vary the free parameter by about 0.05.).

Which of the stationary points is stable and unstable?

Definition: A fixed point (FP) or stationary point of a differentiable function of one variable is a point on the graph of the function where the function's derivative is zero.

According to the definition for stationary points

$$n = n^* \text{ stationary point of } n = n(\tau) \quad \iff \quad f(n^*) := \left. \frac{dn}{d\tau} \right|_{n=n^*} = 0 \quad (10)$$

we have to determine the zero points of the dimensionless equation for population dynamics:

$$f(n) = \frac{dn}{d\tau} = 0 \quad \xleftrightarrow{\text{Eq. (9)}} \quad 0 = Dn - \frac{DA}{K} n^2 - \frac{1}{1+n^{-2}} \quad \text{with} \quad D := \frac{rA}{B} > 0 \quad (11)$$

For $n \neq 0$ we can transform the above equation into a cubic equation:

$$0 \stackrel{!}{=} Dn - \frac{DA}{K} n^2 - \frac{1}{1+n^{-2}} \quad | \cdot (1+n^{-2}) \quad (12)$$

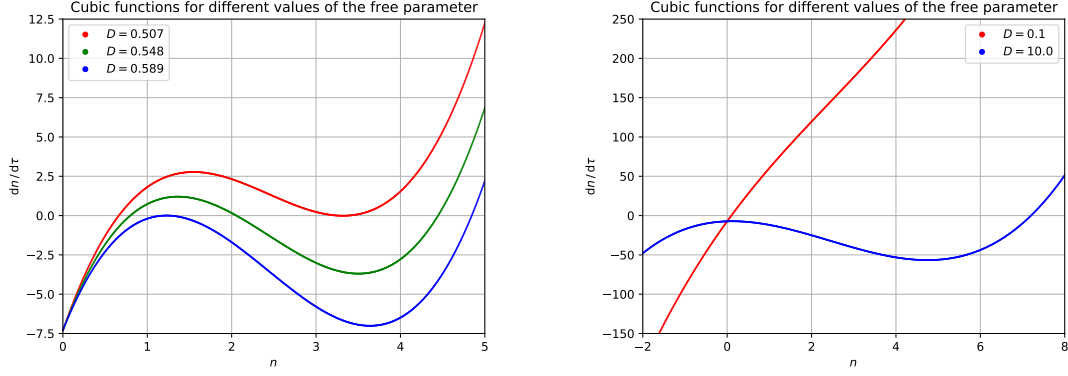
$$\iff \quad = D(1+n^{-2})n - \frac{DA}{K} (1+n^{-2})n^2 - 1 \quad (13)$$

$$\iff \quad = D \left(\frac{n^2+1}{n} \right) - \frac{DA}{K} (n^2+1) - 1 \quad | \cdot n \quad (14)$$

$$\iff \quad = D(n^2+1) - \frac{DA}{K} (n^3+n) - n \quad (15)$$

$$\iff \quad = -\frac{DA}{K} n^3 + Dn^2 - \frac{DA}{K} n - n + D \quad (16)$$

$$\iff \quad = -\frac{DA}{K} n^3 + Dn^2 - \left(\frac{DA}{K} + 1 \right) n + D \quad (17)$$



(a) Cubic functions with $D \in [0.507, 0.589]$ and three stationary points n^* . $D = 0.548$ is the mean of the two interval limits. (b) Cubic function with one stationary point n^* for a small and a large value of the free parameter D .

Figure 1: Cubic functions for different values of the free parameter $D = rA/B > 0$

Finally, we multiply the entire equation by $-K/(DA)$. So we get that for $n \neq 0$ the stationary points n^* are solutions of a cubic equation:

$$f(n) = 0 \quad \Longleftrightarrow \quad 0 \stackrel{!}{=} n^3 - \frac{K}{A} n^2 + \left(1 + \frac{1}{D} \frac{K}{A}\right) n - \frac{K}{A} \quad (18)$$

If we set $K/A = 7.3$ as required in the task, the cubic equation above only depends on n and the remaining free parameter $D = rA/B$:

$$0 \stackrel{!}{=} n^3 - 7.3 n^2 + \left(1 + \frac{7.3}{D}\right) n - 7.3 \quad (19)$$

Let's have a look at the asymptotic behavior of the function:

$$\lim_{n \rightarrow -\infty} f(n) = \lim_{n \rightarrow -\infty} \left[n^3 - 7.3 n^2 + \left(1 + \frac{7.3}{D}\right) n - 7.3 \right] = -\infty \quad (20)$$

$$\lim_{n \rightarrow +\infty} f(n) = \lim_{n \rightarrow +\infty} \left[n^3 - 7.3 n^2 + \left(1 + \frac{7.3}{D}\right) n - 7.3 \right] = +\infty \quad (21)$$

Due to the sign change in the asymptotic behavior for $n \rightarrow \pm\infty$, the function $f(n)$ has at least one real zero point n^* for every value of the parameter $D = rA/B$. We use the following procedure to determine the zeros numerically using **Python**:

- We first determine a zero point using the **SciPy** function **brentq**. The **brentq** method finds a zero point of a given function in a bracketing interval using Brent's method.
- If we know a zero point of the cubic equation, we can use polynomial division to transform the cubic equation into a quadratic equation.
- The zero points of the quadratic equation can now be calculated using the quadratic formula.

Using the quadratic formula and the coefficients of the quadratic equation, we can directly determine the number of real zero points n^* . This allows us to vary the free parameter D (starting at the value $D = 0.01$, step size 10^{-4}) and to see for which values of D one or three real solutions to the function exist (see Python-Code 1). That's how we find out:

- For $D \in [0.507, 0.589]$, the cubic equation in Eq. (19) has exactly three real solutions.
- For $0 < D < 0.507$ or $D > 0.589$, the cubic equation in Eq. (19) has one real solution.

This means that we get three stationary points n^* for $D \in [0.507, 0.589]$ and one stationary point n^* for $0 < D < 0.507$ or $D > 0.589$. Fig. 1 shows the cubic functions for different values of the free parameter $D = rA/B > 0$. The respective stationary points n^* are shown in Table 2.

Table 2: Stationary points and their stability for different values of the free parameter

free paramter D	stationary points n^*		
0.1	0.1 (unstable)		
0.507	0.663 (unstable)	3.261 (stable)	3.376 (unstable)
0.548	0.799 (unstable)	2.055 (stable)	4.446 (unstable)
0.589	1.207 (unstable)	1.248 (stable)	4.845 (unstable)
10.0	7.201 (unstable)		

In order to check the stability of the stationary points n^* , we have to look at the first derivative $f'(n)$ of the function $f(n)$:

$$f'(n) = \frac{df(n)}{dn} = 3n^2 - 2 \frac{K}{A} n + \left(1 + \frac{1}{D} \frac{K}{A}\right) \quad (22)$$

If we use $K/A = 7.3$ again, we get:

$$f'(n) = \frac{df(n)}{dn} = 3n^2 - 14.6 n + \left(1 + \frac{7.3}{D}\right) \quad (23)$$

Now we can determine the stability of the stationary points n^* shown in Table 2. According to the definition from the lecture we get:

$$f'(n^*) < 0 \quad \implies \quad \text{the stationary point } n^* \text{ is stable} \quad (24)$$

$$f'(n^*) > 0 \quad \implies \quad \text{the stationary point } n^* \text{ is unstable} \quad (25)$$

We noted the stability of the individual stationary points n^* in Table 2. Based on the results, we assume that in the case of three stationary points n^* , the middle stationary point is always stable and the other two are each unstable.

Python-Code 1: Numerical determination of the stationary points

```

1  # -*- coding: utf-8 -*-
2  """
3  Introduction to Computational Physics
4  - Exercise 07: Population Dynamics - Stationary Points
5  - Group: Simon Groß-Bölting, Lorenz Vogel, Sebastian Willenberg
6  """
7
8  import numpy as np; import matplotlib.pyplot as plt
9  from scipy.optimize import brentq
10
11 def cubic_function(n, coefficients):
12     ''' Cubic function where n represents an unknown (variable)
13         and a, b, c and d represent known numbers (coefficients) '''
14     a,b,c,d = coefficients[0], coefficients[1], coefficients[2], coefficients[3]
15     return (a*n**3)+(b*n**2)+(c*n)+d
16
17 def quadratic_formula(coefficients):
18     ''' Function to compute the solutions of a reduced quadratic equation
19         using the quadratic formula '''
20     p = coefficients[1]/coefficients[0]
21     q = coefficients[2]/coefficients[0]
22     if ((p/2)**2 >= q): # as stationary points only real solutions are valid
23         n2 = -(p/2)+np.sqrt((p/2)**2-q)
24         n3 = -(p/2)-np.sqrt((p/2)**2-q)
25         return (n2,n3)
26     else: return None
27
28 def count_zeros(coefficients):
29     ''' Function to compute the number of real zeros of a quadratic equation
30         Input: NumPy-Array with the coefficients of the quadratic equation
31         Output: number of real zeros of the given quadratic equation '''
32     p = coefficients[1]/coefficients[0]
33     q = coefficients[2]/coefficients[0]
34     if ((p/2)**2 < q):
35         # in this case the square root becomes imaginary and
36         # the quadratic equation has no real zero points
37         return 0
38     elif ((p/2)**2 == q):
39         # in this case the square root becomes zero and
40         # the quadratic equation has a real 'double zero point'
41         return 1
42     else: return 2
43
44 def find_solutions(D,D_step):
45     ''' Function to answer the question: When do one or three real solutions
46         exist as a function of the remaining free parameter?
47         We vary the free parameter D and check for which value of D we
48         get one or three real solutions
49         Input: starting value for the free paramter D '''
50
51     # We use the coefficients of the cubic equation, which
52     # we have already derived analytically
53     coefficients = np.array([1., -7.3, 1+(7.3/D), -7.3])
54
55     # compute first zero point of the cubic equation numerically
56     n1 = brentq(cubic_function, -2., 8., args=coefficients)
57
58     # compute the quadratic function using polynomial division
59     quadratic_function = np.polydiv(coefficients, np.array([1., -n1]))[0]
60
61     D_lower = D; D_upper = D
62     while count_zeros(quadratic_function) != 2:
63         D += D_step # increase free paramter
64         coefficients = np.array([1., -7.3, 1+(7.3/D), -7.3])
65
66         # compute first zero point of the cubic equation numerically
67         n1 = brentq(cubic_function, -2., 8., args=coefficients)
68
69         # compute the quadratic function using polynomial division
70         quadratic_function = np.polydiv(coefficients, np.array([1., -n1]))[0]
71         D_lower = D
72

```

```

73 while count_zeros(quadratic_function) == 2:
74     D_upper = D; D += D_step # increase free parameter
75     coefficients = np.array([1., -7.3, 1+(7.3/D), -7.3])
76
77     # compute first zero point of the cubic equation numerically
78     n1 = brentq(cubic_function, -2., 8., args=coefficients)
79
80     # compute the quadratic function using polynomial division
81     quadratic_function = np.polydiv(coefficients, np.array([1., -n1]))[0]
82
83     return (D_lower, D_upper)
84
85 def calculate_zeros(D):
86     coefficients = np.array([1., -7.3, 1+(7.3/D), -7.3])
87
88     # compute first zero point of the cubic equation numerically
89     n1 = brentq(cubic_function, -2., 8., args=coefficients)
90
91     # compute the quadratic function using polynomial division
92     quadratic_function = np.polydiv(coefficients, np.array([1., -n1]))[0]
93     n2, n3 = quadratic_formula(quadratic_function)
94     return (n1, n2, n3)
95
96
97 # We determine the stationary points for K/A = 7.3
98 # The stationary points are solutions of a cubic equation; it depends on
99 # the variable n and the remaining free parameter D
100 fig1, ax1 = plt.subplots(); fig2, ax2 = plt.subplots()
101
102 D = find_solutions(0.01, 1e-4)
103 n = np.linspace(0, 5, 1000)
104 coefficients = np.array([1., -7.3, 1+(7.3/D[0]), -7.3])
105 ax1.plot(n, cubic_function(n, coefficients), 'r.', markersize=1,
106         label=r'$D={D}$'.format(D=round(D[0], 3)))
107 coefficients = np.array([1., -7.3, 1+(7.3/(0.5*(D[0]+D[1]))), -7.3])
108 ax1.plot(n, cubic_function(n, coefficients), 'g.', markersize=1,
109         label=r'$D={D}$'.format(D=round(0.5*(D[0]+D[1]), 3)))
110 coefficients = np.array([1., -7.3, 1+(7.3/D[1]), -7.3])
111 ax1.plot(n, cubic_function(n, coefficients), 'b.', markersize=1,
112         label=r'$D={D}$'.format(D=round(D[1], 3)))
113
114 # print zero points for different values of the free parameter
115 for D in [D[0], 0.5*(D[0]+D[1]), D[1]]:
116     n1, n2, n3 = calculate_zeros(D)
117     zeros = np.array([round(n1, 3), round(n2, 3), round(n3, 3)])
118     print('stationary points for D={}: {}'.format(round(D, 3), zeros))
119 for D in [0.1, 10]:
120     coefficients = np.array([1., -7.3, 1+(7.3/D), -7.3])
121     n1 = brentq(cubic_function, -2., 8., args=coefficients)
122     print('stationary points for D={}: {}'.format(round(D, 3), round(n1, 3)))
123
124 D = np.array([0.1, 10]) # set the free parameter to a small and a large value
125 n = np.linspace(-2, 8, 1000)
126 coefficients = np.array([1., -7.3, 1+(7.3/D[0]), -7.3])
127 ax2.plot(n, cubic_function(n, coefficients), 'r.', markersize=1,
128         label=r'$D={D}$'.format(D=round(D[0], 3)))
129 coefficients = np.array([1., -7.3, 1+(7.3/D[1]), -7.3])
130 ax2.plot(n, cubic_function(n, coefficients), 'b.', markersize=1,
131         label=r'$D={D}$'.format(D=round(D[1], 3)))
132
133 for ax in [ax1, ax2]:
134     ax.set_title('Cubic functions for different values of the free parameter')
135     ax.set_xlabel(r'$n$'); ax.set_ylabel(r'$\mathrm{d}n\backslash,\backslash,\mathrm{d}\backslash\tau$')
136     ax.grid(); ax.legend(loc='best', markerscale=8)
137 ax1.set_xlim((0, 5)); ax1.set_ylim((-7.5, 12.5))
138 ax2.set_xlim((-2, 8)); ax2.set_ylim((-150, 250))
139 fig1.savefig('figures/Stationary-Points-01.pdf', format='pdf')
140 fig2.savefig('figures/Stationary-Points-02.pdf', format='pdf')
141 plt.show(); plt.clf(); plt.close()

```