

Pharmacokinetics Modeling Course

Ordinary Differential Equations (ODE) & Compartment Models




Dr. Matthias König
 Humboldt-University Berlin
 Systems Medicine of the Liver
koenigmx@hu-berlin.de
<https://livermetabolism.com>

By the end of this section, you should be able to:

1. **Understand the role of ODEs** in pharmacokinetic modeling to describe dynamic changes in drug concentration over time.
2. **Interpret ODEs** used in simple compartment models.
3. Gain **basic intuition for numerical integration methods**, such as Euler's method methods.
4. **Apply numerical integration techniques** to simulate drug concentration-time profiles.
5. **Understand how ODE-based models are implemented** in simulation tools and programming environments (e.g., Python).

Ordinary differential equations (ODE)

- 
- a differential equation describes the **rate of change of a variable**
 - **dC/dt** denotes the rate of change of the concentration over time
 - differential equations require specification of the **initial value (C₀)**
 - solution
 - Normally no analytic solution
 - Numerically approximation

$$\frac{dC}{dt} = -\frac{CL}{V} * C$$
$$C_0 = \frac{\text{Dose}}{V}$$

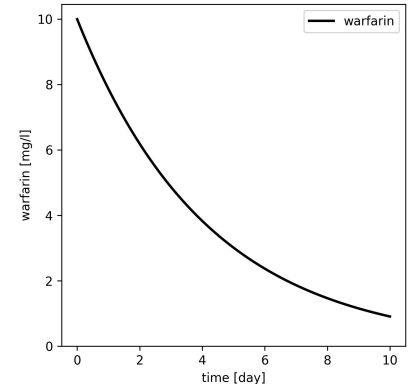
Ordinary differential equations (ODE)

- a differential equation describes the **rate of change of a variable**
- **dC/dt** denotes the rate of change of the concentration over time
- differential equations require specification of the **initial value (C₀)**
- solution
 - Normally no analytic solution
 - Numerically approximation

$$\frac{dC}{dt} = -\frac{CL}{V} * C$$
$$C_0 = \frac{\text{Dose}}{V}$$

analytic solution

$$C_{(t)} = \frac{\text{Dose}}{V} e^{-\frac{CL}{V} \cdot t}$$



Mould DR, Upton RN. **Basic concepts in population modeling, simulation, and model-based drug development.** CPT Pharmacometrics Syst Pharmacol. 2012 Sep 26;1(9):e6. doi: 10.1038/psp.2012.4. PMID: 23835886; PMCID: PMC3606044.

Euler Method

Euler method is the simplest **numerical method to solve ODEs**

Ordinary differential equation

$$\frac{d\vec{x}}{dt} = \vec{f}(\vec{x}, \vec{p}, t)$$

Single state variable, initial condition

$$\frac{dx}{dt} = f(x) \quad \text{with} \quad x(t=0) := x_0$$

1. Approximate rate of change

$$f(x) = \frac{dx}{dt} \approx \frac{x(t + \Delta t) - x(t)}{\Delta t}$$

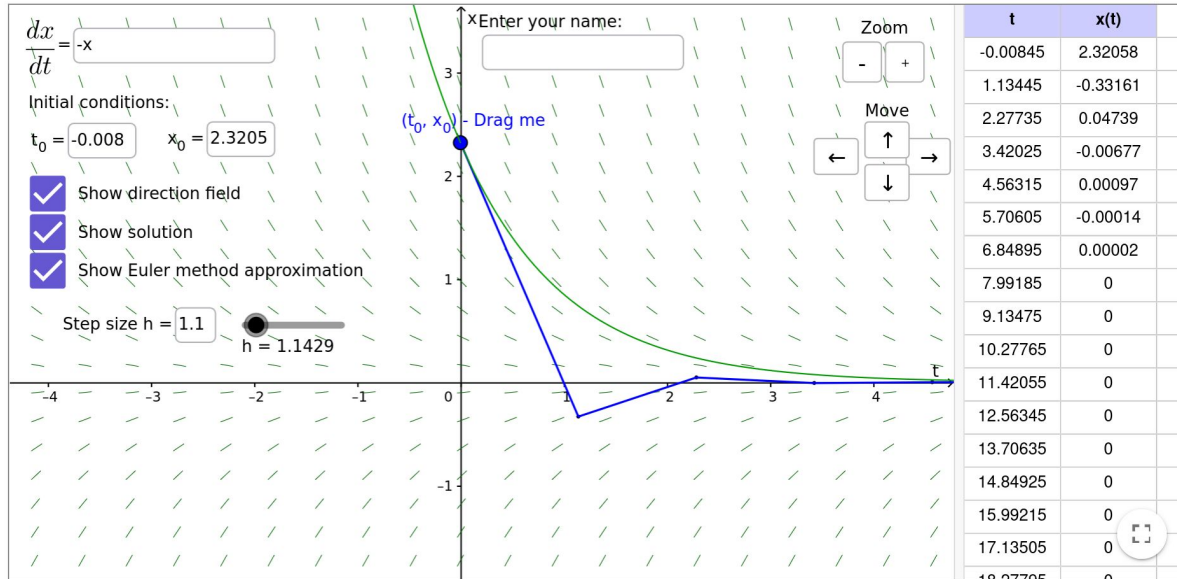
2. Get next solution value

$$x(t + \Delta t) = x(t) + \Delta t f(x(t)) + O(\Delta t^2)$$

3. Goto 1

Euler Method

<https://melbapplets.ms.unimelb.edu.au/2021/07/08/exploring-an-ode-with-euler-method/>



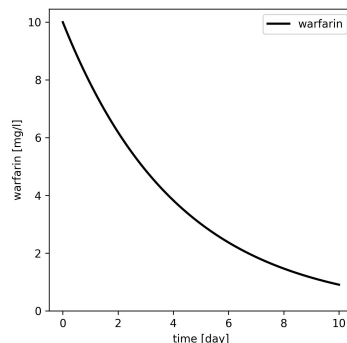
$$\frac{dC}{dt} = -\frac{CL}{V} * C$$

$$C_0 = \frac{\text{Dose}}{V}$$

Numerical integration in Python

- ODEs can be solved via numerical integration
- e.g. Euler method as simplest case
- solving a system of equations is computationally expensive

$$\frac{dC}{dt} = -\frac{CL}{V} * C$$
$$C_0 = \frac{Dose}{V}$$



```
1  from scipy.integrate import odeint
2  from matplotlib import pyplot as plt
3  import numpy as np
4
5  # Parameter
6  V = 10 # [l]
7  CL = 0.1 # [L/hr]
8  Dose = 100 # [mg]
9
10
11 new *
12 def ydot(y, t):
13     """ODE system: dx/dt"""
14     C = y[0]
15     return np.array([-CL/V * C])
16
17 # initial condition
18 y0 = np.array([Dose/V, ]) # [mg/l]
19
20 # Numerical integration
21 t = np.linspace(start=0, stop=10*24, num=200) # [hr]
22 C = odeint(ydot, y0, t)
23
24 f, ax = plt.subplots(nrows=1, ncols=1, figsize=(5, 5), dpi=300)
25 ax.plot(t/24.0, C[:, 0], label="warfarin", color="black", linewidth=2.0)
26 ax.set_xlabel("time [day]")
27 ax.set_ylabel("warfarin [mg/l]")
28 ax.set_ylim(bottom=0)
29 ax.legend()
30 plt.show()
```

Compartment models

- Pharmacokinetics can be modeled via compartment models
- Simple pharmacokinetic models have proven useful in many applications
- Main processes (**ADME**)
 - Absorption
 - Distribution
 - Metabolization
 - Excretion

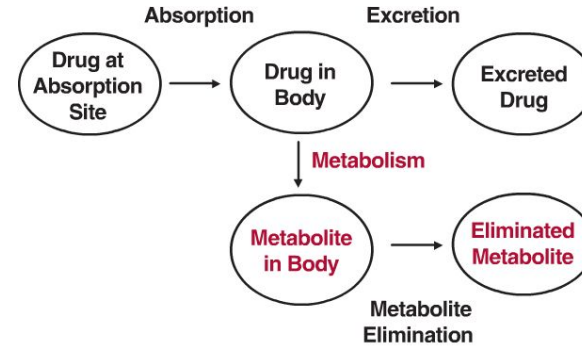


FIGURE 2-5. A drug is simultaneously absorbed into the body and eliminated from it, by excretion and metabolism. The processes of absorption, excretion, and metabolism are indicated with arrows and the compartments with ovals. The compartments represent different locations and different chemical species (color = metabolite). Metabolite elimination may occur by further metabolism or excretion.

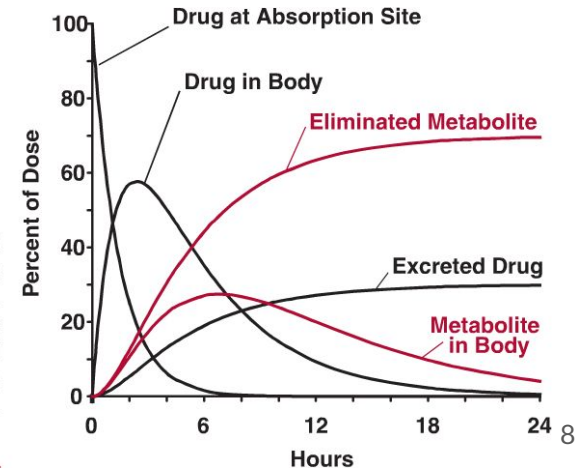


FIGURE 2-6. Time course of drug and metabolite in each of the compartments shown in Fig. 2-5. The amount in each compartment is expressed as a percentage of the dose administered. In this example, all the dose is absorbed. At any time, the sum of the molar amounts in the five compartments equals the dose.

Example of compartment model

- system of ODEs
- Solved numerically
- A(D)ME
 - Absorption (v_a)
 - Metabolism (v_m)
 - Elimination ($v_{u,A}, v_{u,B}$)
- Mass action equations with rate constants k_a, k_m, k_e

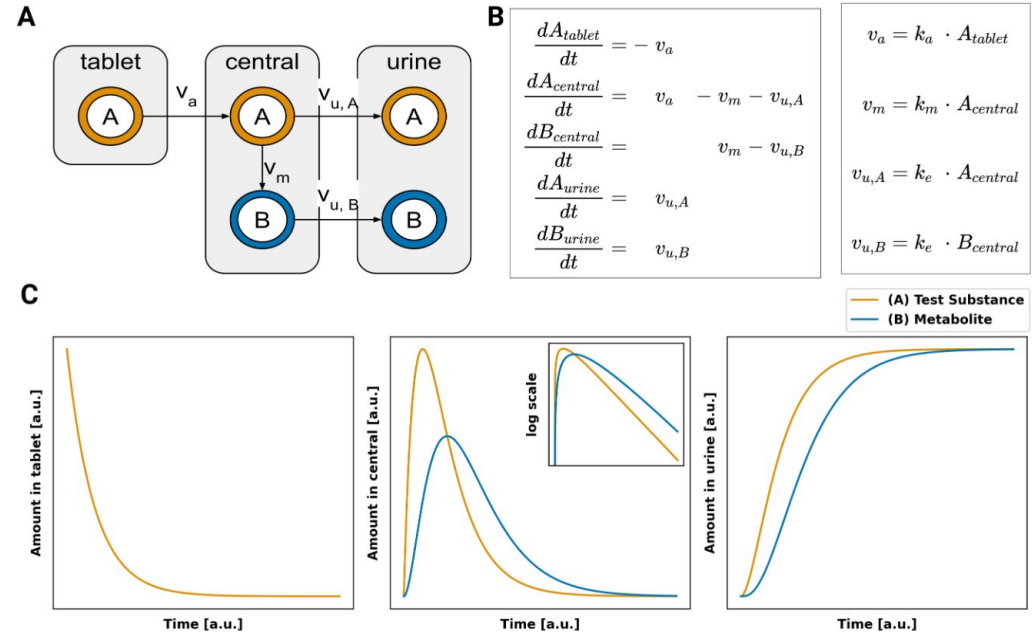


FIGURE 2.3: **Simple ODE-based pharmacokinetics model.** **A)** The system consists of three compartments (tablet, central, urine) that are connected via transport reactions. The model contains two substances the test substance A (orange); and the metabolite B (blue). The test substance A is metabolized to metabolite B in the central compartment. **B)** The resulting system of ordinary differential equations (ODEs). The rate of absorption, metabolism, and excretion ($v_a, v_m, v_{u,A}, v_{u,B}$) are modeled via irreversible mass-action kinetics. **C)** With an initial amount of $A_{\text{tablet}} = 10$ and rates $k_a = 1, k_m = 1$, and $k_e = 1$, all in [a.u.], the resulting amounts over time of the substances in the tablet, central, urine compartments are depicted.

Physiologically based pharmacokinetic (PBPK) modeling for dynamical liver function tests and CYP phenotyping. Jan Grzegorzewski (supervisor: **Matthias König**). PhD Thesis, Jan 2023



The
Scientific
Python
Development
Environment

A screenshot of the Spyder Python IDE interface. The interface is divided into several panels. On the left is a file explorer showing a project structure with folders like 'Plots' and 'plugin.py'. The central panel displays a Python script named 'plugin.py' with code for a 'Plots Plugin'. The right panel shows a 'Variable Explorer' with a table of variables and their values. At the bottom right, there is a 'Plots' panel displaying a 3D surface plot and a corresponding polar plot. The status bar at the very bottom shows the current file, line, column, and other details.

plugin.py

- Plots
 - get_name
 - get_description
 - get_icon
 - register
 - unregister
 - switch_to_plugin
 - current_widget
 - add_shellwidget
 - remove_shellwidget
 - set_shellwidget
- plot_example.py
 - Unnamed cell 1
 - generate_polar_plot
 - generate_dem_plot
 - main
- plugin.py
 - IPythonConsole

```
1  # -*- coding: utf-8 -*-
2
3  # Copyright © Spyder Project Contributors
4  # Licensed under the terms of the MIT License
5  # (see spyder/__init__.py for details)
6
7  """
8  Plots Plugin.
9  """
10
11 # Third party imports
12 from qtpy.QtCore import Signal
13
14 # Local imports
15 from spyder.api.plugins import Plugins, SpyderDockablePlugin
16 from spyder.api.translations import get_translation
17 from spyder.plugins.plots.widgets.main_widget import PlotsWidget
18
19 # Localization
20 _ = get_translation('spyder')
21
22 class Plots(SpyderDockablePlugin):
23     """
24     Plots plugin.
25     """
26
27     NAME = 'plots'
28     REQUIRES = (Plugins.IPythonConsole)
29     TABIFY = (Plugins.VariableExplorer, Plugins.Help)
30     WIDGET_CLASS = PlotsWidget
31     CONF_SECTION = NAME
32     CONF_FILE = False
33     DISABLE_ACTIONS_WHEN_HIDDEN = False
34
35 # -- SpyderDockablePlugin API
36 # --
37 def get_name(self):
38     return _('Plots')
39
40 def get_description(self):
41     return _('Display, explore and save console generated plots.')
42
43 def get_icon(self):
44     return self.create_icon('hist')
45
46 def register(self):
47     # Plugins
48     ipyconsole = self.get_plugin(Plugins.IPythonConsole)
49
50     # Signals
51     ipyconsole.sig_shellwidget_changed.connect(self.set_shellwidget)
52     ipyconsole.sig_shellwidget_created.connect(self.add_shellwidget)
53     self.add_shellwidget()
54     ipyconsole.sig_shellwidget_deleted.connect(
55         self.remove_shellwidget)
56
```

Name	Type	Size	Value
a	foo	1	foo object of __main__ module
filename	str	53	/Users/juanitagomez/spyder/spyder/tests/test_dont_use.py
i	bool	1	True
my_set	set	3	{1, 2, 3}
r	float	1	0.46567886443
t	tuple	5	('abcd', 745, 2.23, 'efgh', 78.2)
thisdict	dict	3	{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
tinylist	list	2	[123, 'efgh']
x	Array of int64 (2,)		[1 2]
y	timedelta	1	2 days, 0:00:00

Help Variable Explorer Files

97 %

Plots IPython console History

LSP Python: ready conda: spyder-dev, Python 3.7.10, master Line 1, Col 1 UTF-8 LF RW Mem 57%