BIOREACTORS MODELING AND SIMULATION

ChE-320

Professor: Dr. Vassily Hatzimanikatis

ASSISTANTS

- Denis Joly denis.joly@epfl.ch
- Ilias Toumpe ilias.toumpe@epfl.ch
- Omar Keshk omar.keshk@epfl.ch
- Yigit Sibal yigit.sibal@epfl.ch

OBJECTIVE

- Explore and analyze different bioprocesses to understand:
  - Their basic mechanisms and parameters
  - Their design workflow
  - Optimal operation

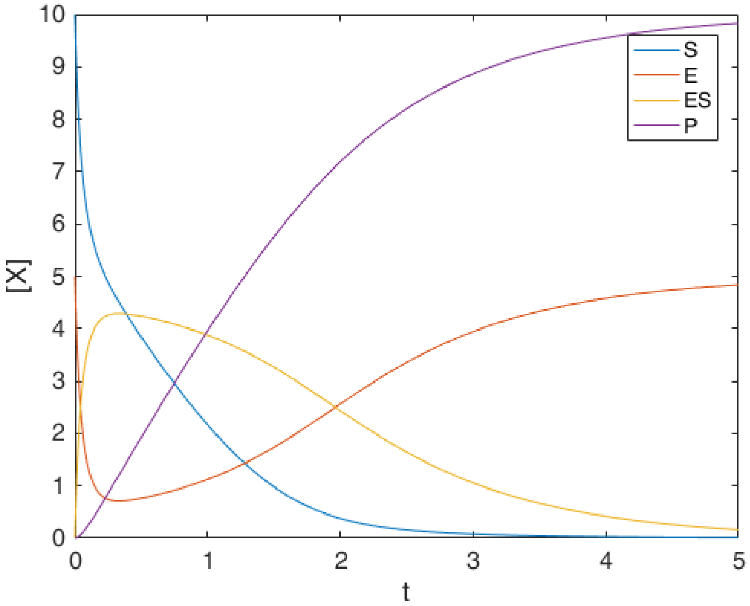
- Develop relevant programming skills:
  - Write code in an efficient helpful structure
  - Understand how to convert a problem into code
  - Visualize and analyze results

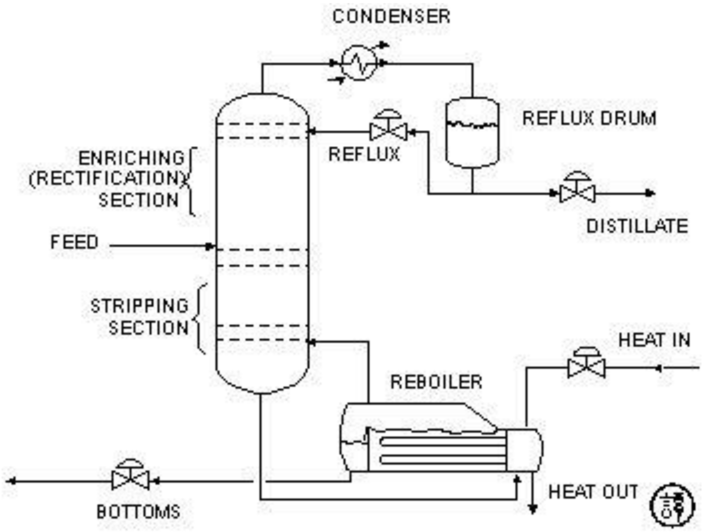
CONTENT

Enzyme and microbial kinetics

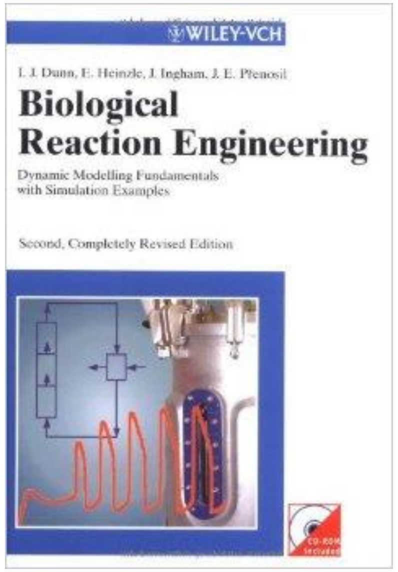
Application of chemical engineering design principles

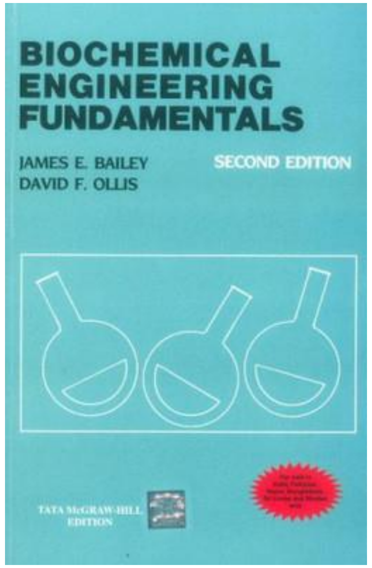
Modeling and simulation of bioreactors





TEXTBOOKS





"Biological Reaction Engineering: Dynamic Modeling Fundamentals with Simulation Examples"
I. J. Dunn, E. Heinzle, J. Ingham, and J. E. Prenosil
Ed. Wiley-Vch

"Biochemical Engineering Fundamentals"
J. E. Bailey and D. F. Ollis
Ed. McGraw-Hill Science

COURSE PLAN

Sessions	Content
17 February – 25 February	Introduction to programming
3 March – 11 March	PROJECT 1
17 March – 1 April	PROJECT 2
7 April – 29 April	PROJECT 3
17 April – 27 April	Spring break
5 May– 13 May	PROJECT 4
19 May – 27 May	PROJECT 5

PROJECT REPORTS

- Deliverables:
  - 1 report (per group) in PDF format
  - All the code files required for replicating the results of the report

- Files are uploaded through the Moodle course page

- The deadline for delivery of each project is at **23h59 before** the start of the next project

PROJECT REPORTS

- Deliverables:
  - 1 report (per group!) in PDF format
  - All the code files required for replicating the results of the report

- The report:
  - Maximum of 9 pages (any extra pages will not be graded!)
  - Title page with names of group members and Project title
  - Introduction discussing the goals of the Project
  - Discussion of the results and diagrams for each question separately
  - Conclusion summarizing the key findings for each exercise

PROJECT REPORTS

- Deliverables:
  - 1 report (per group!) in PDF format
  - All the code files required for replicating the results of the report

- The code:
  - Can be one or more files that are clearly named (e.g. exercise_1.py, excerise_2.py)
  - Should contain comments explaining the variables, the steps followed, or the inputs and outputs of a function
  - **Important:** The code should run by simply executing the script/main function

GRADING

Activity	Points
Exercises	5/6
Code format, Clarity of presentation of results	1/6
Bonus questions (optional)	+0.1 per project

Grades will be uploaded in Moodle within two weeks after submission.

GROUPS

- Send your groups to:

Denis Joly denis.joly@epfl.ch

- Groups of 4-5 students
- Deadline: February 28th

A SYSTEM OF ODES

- A set of equations that includes derivatives
- Only one independent variable

Example: damped harmonic oscillator

$$\begin{cases} \dfrac{dy}{dt} = z \\ \dfrac{dz}{dt} = -2\gamma\omega\, z - \omega^2 y \end{cases}$$

Variables: y, z
Parameters: γ, ω

A SYSTEM OF ODEs

- A set of equations that includes derivatives
- Only one independent variable

Example: damped harmonic oscillator

$$\begin{cases} \dfrac{dy}{dt} = z \\ \dfrac{dz}{dt} = -2\gamma\omega\, z - \omega^2 y \end{cases}$$

$$\Leftrightarrow \frac{d}{dt}\begin{bmatrix} y \\ z \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega^2 & -2\gamma\omega \end{bmatrix} \cdot \begin{bmatrix} y \\ z \end{bmatrix}$$

Variables: y, z

Parameters: γ, ω

⇔ first order system of the form

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

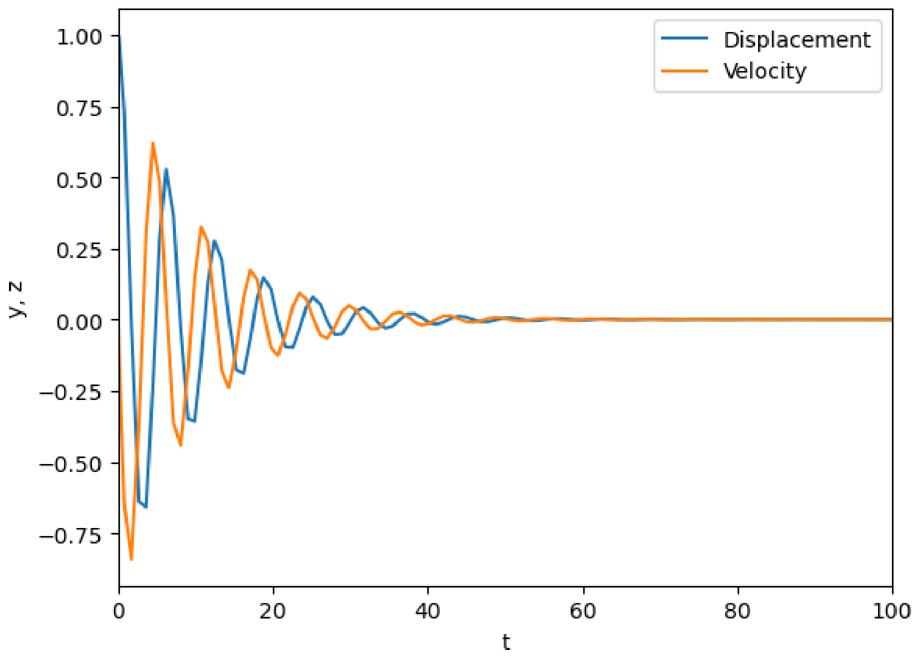
⇔ Matrix formulation (**only** Linear ODE)

How to PYTHON this

```python
# Import libraries
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp


# Add parameters
w = 1
g = 0.1

# Define the function for the derivatives
def dX_dt(t, X):
    """
    Compute the derivatives for the damped harmonic oscillator.

    Parameters:
    t (float): Time variable.
    X (array): State vector [y, z], where y is displacement and z is velocity.

    Returns:
    list: Derivatives [dy/dt, dz/dt] = [z, -w^2 * y - 2 * w * g * z].
    """
    return [X[1], -w**2 * X[0] - 2 * w * g * X[1]]
```

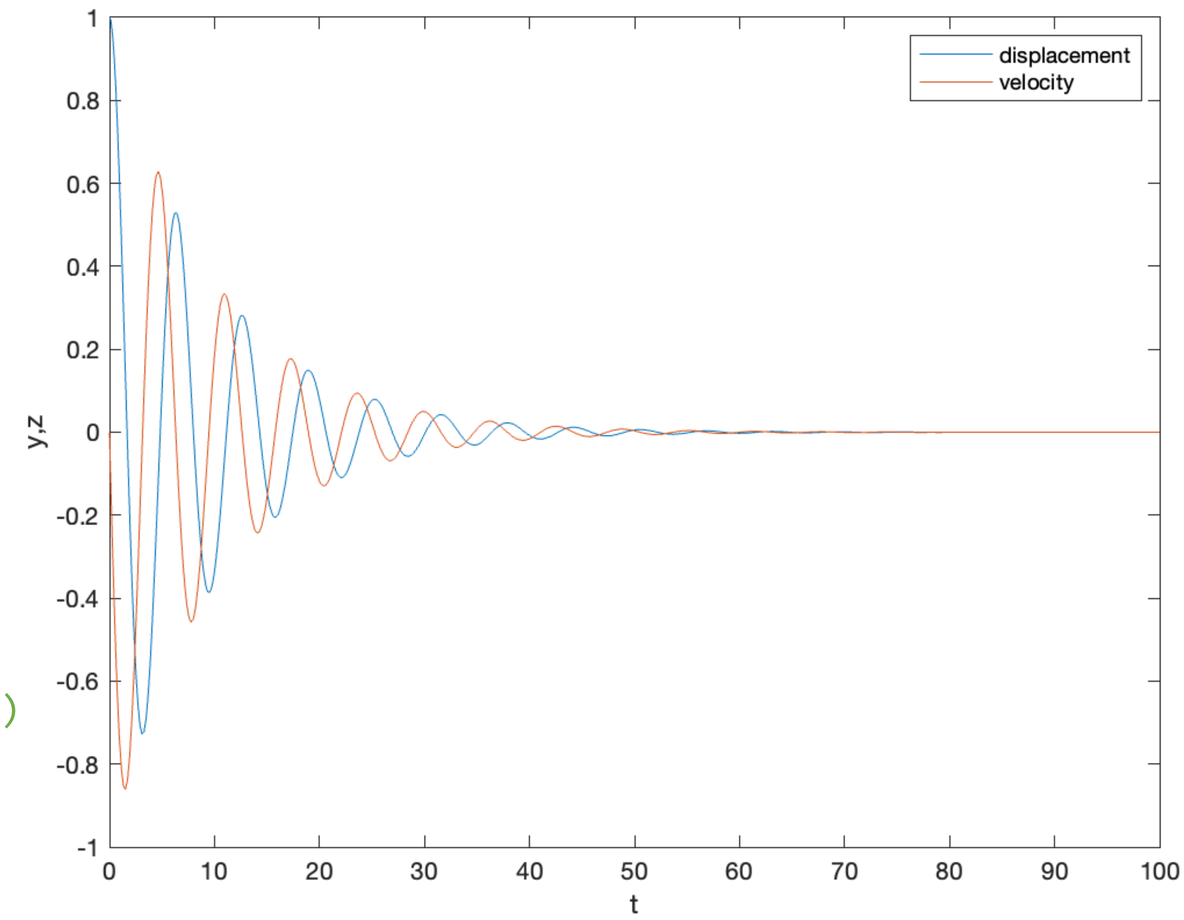
How to PYTHON this

```python
# Initial conditions
X0 = [1, 0]
time_int = [0, 100]

# Solving ODE:
# solve_ivp(function, time interval, initial condition, method)
sol = solve_ivp(dX_dt, time_int, X0)

# Plotting
plt.plot(sol.t, sol.y[0], label='Displacement')
plt.plot(sol.t, sol.y[1], label='Velocity')
plt.xlabel('t')
plt.ylabel('y, z')
plt.xlim([-0.001, 100])
plt.legend()
plt.show()
```

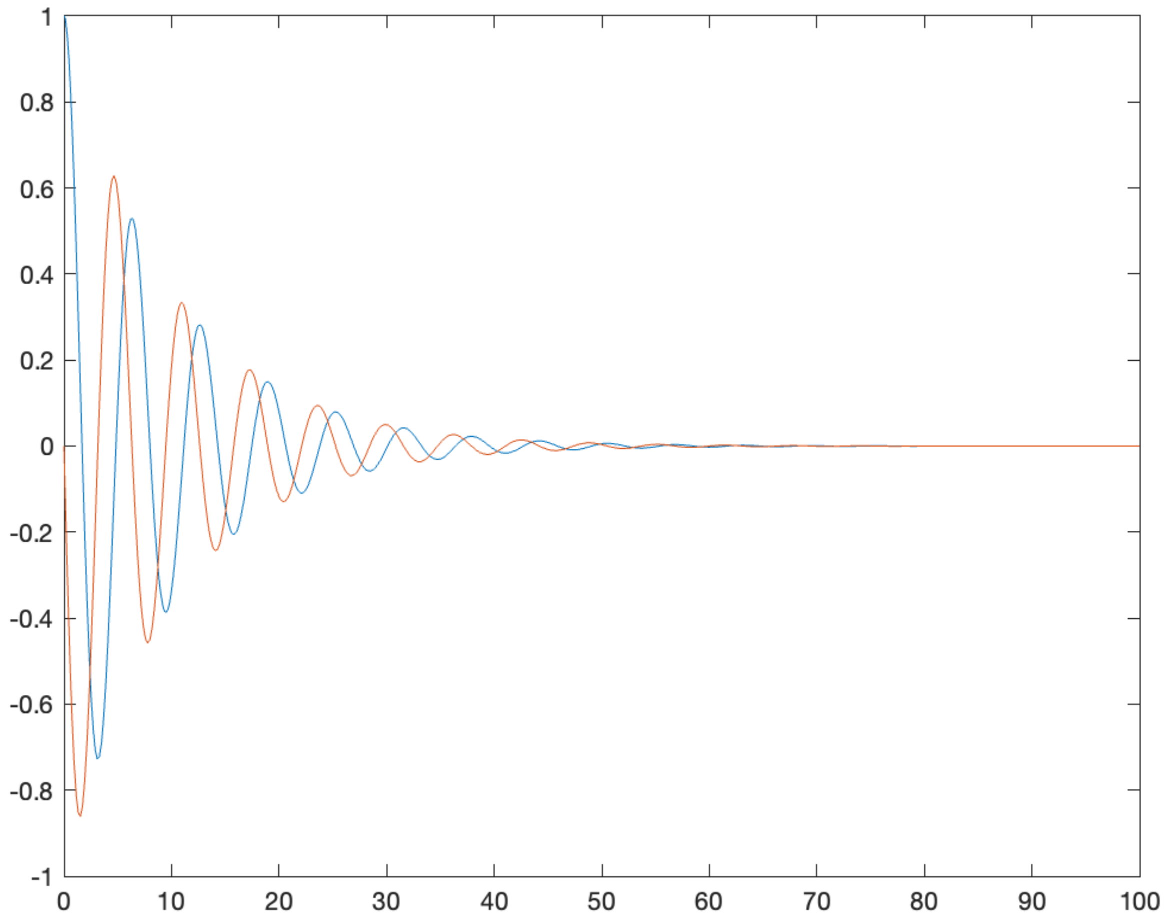


How to MATLAB this

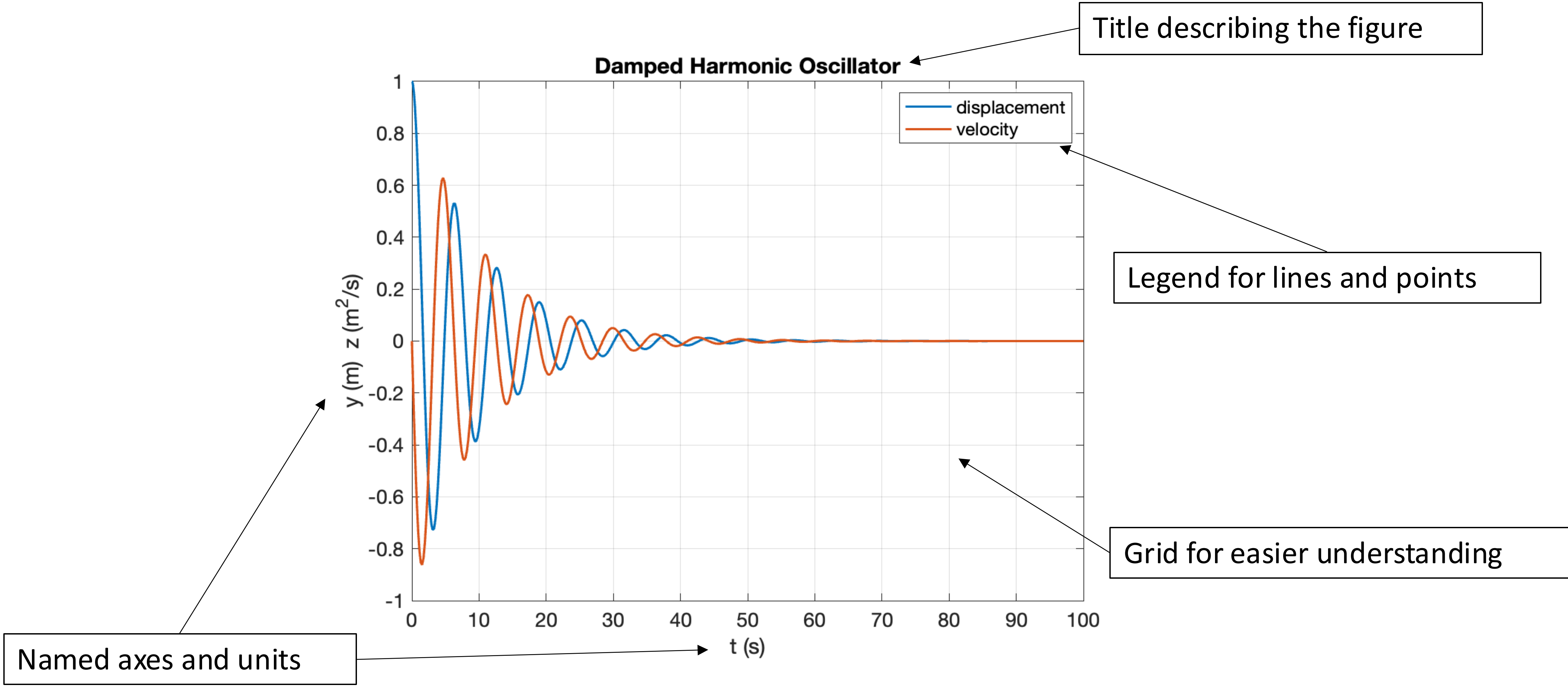
```
% Parameters
w = 1;
g = 0.1;

% X = [y;z], w = omega, g = gamma
f = @(t,X) [X(2);-w^2*X(1)-2*w*g*X(2)]

% Initial_conditions
X0 = [1;0];
time_int = [0,100];

% Solving ODE:
% ode45(function handle, time interval, initial condition)
[t,X] = ode45(f,time_int,X0);

plot(t,X);
xlabel('t'), ylabel('y,z'), legend('displacement', 'velocity')
```



HOW TO MAKE A USEFUL PLOT

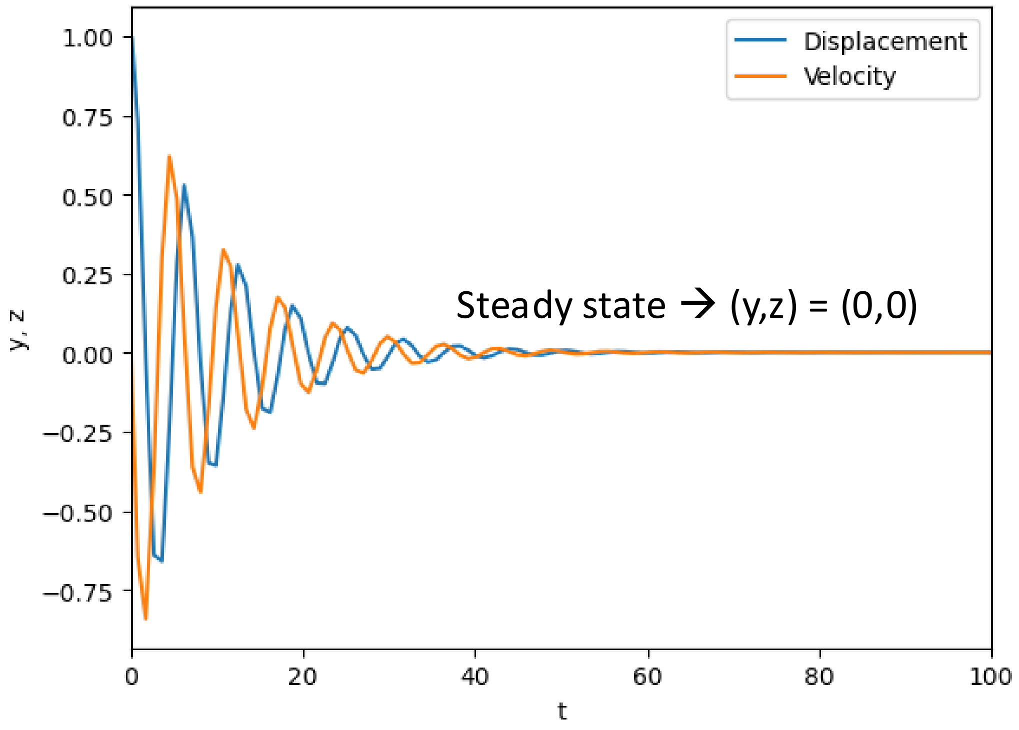


How to make a useful plot



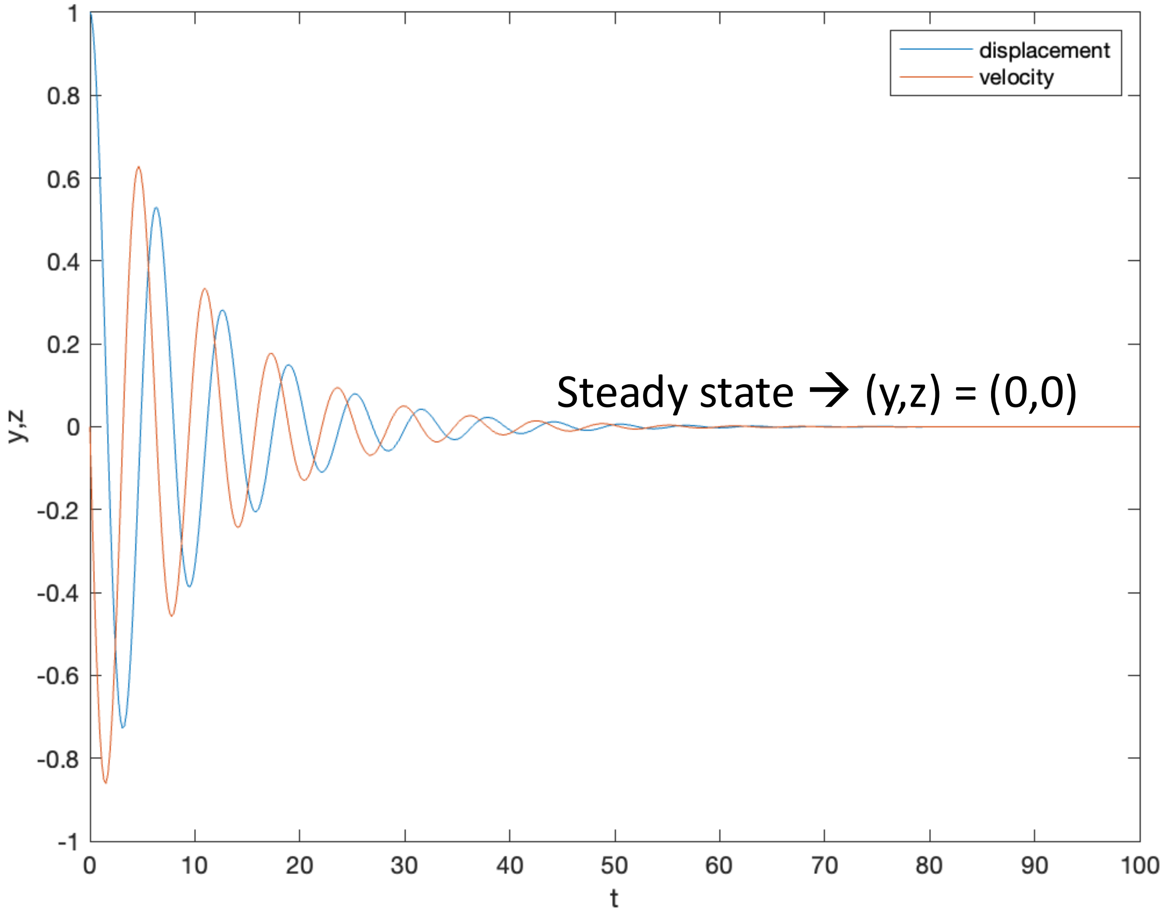
STEADY STATE VALUES

```python
# Import libraries
import numpy as np
from scipy.optimize import fsolve

# Add parameters
w = 1
g = 0.1

# Define the function for the derivatives
def dX_dt(X):
    """
    Compute the derivatives for the damped harmonic oscillator.

    Parameters:
    X (array): State vector [y, z], where y is displacement and z is velocity.

    Returns:
    list: Derivatives [dy/dt, dz/dt] = [z, -w^2 * y - 2 * w * g * z].
    """
    return [X[1], -w**2 * X[0] - 2 * w * g * X[1]]

# Initial guess
Y0 = np.array([1, 0])

# Solving ODE using fsolve
X_ss = fsolve(dX_dt, Y0)

# Print the steady state
print('The steady state for the displacement and velocity is:', X_ss)
```



STEADY STATE VALUES (MATLAB)

```
% Parameters
w = 1;
g = 0.1;

% X = [y;z], w = omega, g = gamma
% The matlab solver needs f(y)
f = @(X) [X(2);-w^2*X(1)-2*w*g*X(2)]

%Initial_guess
Y0 = [1,0]
% Solving ODE:
% fsolve(function, initial guess)
X_ss = fsolve(f,Y0);
```



Your turn to try!