Computational Techniques and Programming Languages

Course Code: PHY421

Dr. D.V. Senthilkumar

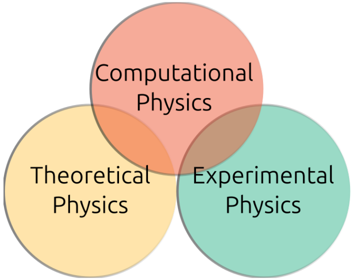
Schoool of Physics
IISER Thiruvananthapuram

January 31, 2024

Introduction

What is Computational Physics?

Computational physics is the study and implementation of numerical analysis to solve problems in physics for which a quantitative theory already exists; it combines computer science, physics and applied mathematics to develop scientific solutions to complex problems.

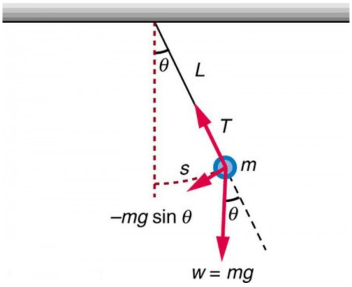


Computational physics will allow you to tackle realistic problems in practically every field of science and engineering.

Analytically Intractable Problems

For demonstration, we consider a generic problem of solving the dynamical equation for simple pendulum.

$$\ddot{\theta} + (g/L)\sin\theta = 0 \tag{1}$$



Analytically Intractable Problems

Even simple problem like this becomes analytically intractable for large θ. The standard analytic approach at this point is to make the further assumption that the **angle of motion is small.** In this limit, we can approximate $\sin\theta \approx \theta$, which gives us a familiar-looking equation:

$$\ddot{\theta} + \left(\frac{g}{L}\right)\theta \approx 0 \tag{2}$$

Adding friction and external drive can futher complicate the problem.

$$\ddot{\theta} + b\dot{\theta} + \left(\frac{g}{L}\right)\sin\theta = f\sin(\omega t + \phi) \tag{3}$$

Therefore, we use numerical approaches to tackle such problems.

Need for Computational Physics

- **Solving complex problems:**
  - Tackle problems that are analytically intractable.
  - Examples: fluid dynamics, molecular simulations, weather prediction.
- **Exploring new phenomena:**
  - Investigate physical systems that are difficult or impossible to access experimentally.
  - Examples: black holes, particle collisions, early universe conditions.
- **Making predictions:**
  - Test theories and make predictions about physical phenomena without relying solely on experiments.
  - Examples: material properties, drug design, climate change projections.
- **Visualizing complex systems:**
  - Create informative visualizations to gain insights and communicate results effectively.
  - Examples: animations, 3D renderings, interactive simulations.

Where Computational Physics is Used?

- **Astrophysics and Cosmology:** Computational simulations model celestial entities—galaxies, stars, planets, and even black holes—to unravel their behaviors.
- **High-Energy Physics:** Employing computational simulations to understand subatomic particle behaviors in accelerators and to decipher fundamental natural forces.
- **Condensed Matter Physics:** Utilizing computational simulations to explore material behaviors at atomic and molecular scales, aiding in material comprehension and design.

Where Computational Physics is Used?

- **Biophysics:** Employing computational simulations to study biological systems like DNA, proteins, cells, and tissues, elucidating their properties and functions.
- **Climate Modeling:** Leveraging computational simulations to comprehend Earth's climate system, assess human-induced impacts on climate, and predict future changes.
- **Engineering and Technology:** Utilizing computational simulations in diverse engineering applications, such as fluid dynamics, structural mechanics, and electromagnetic studies.

Examples

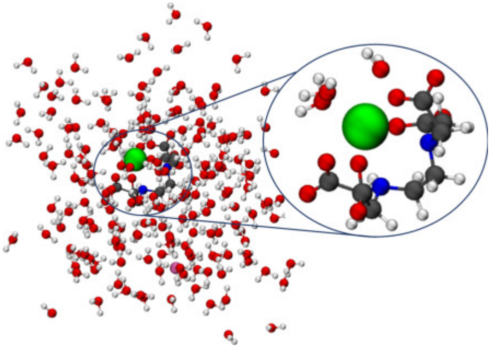


Figure: Molecular Dynamics Simulation

Examples

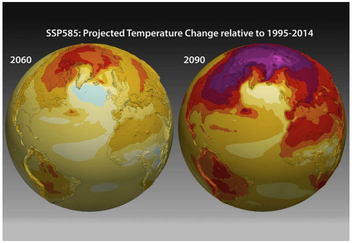


Figure: Climate Simulations

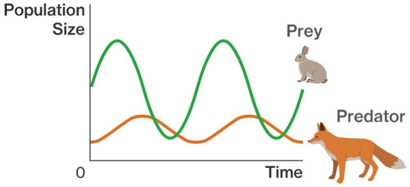
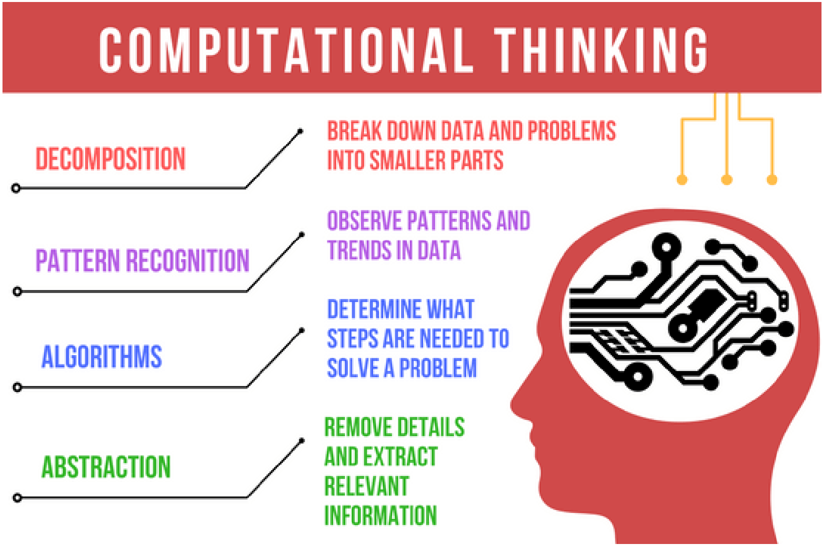


Figure: Predator-Prey Dynamics

Computational Thinking



COMPUTATIONAL THINKING

DECOMPOSITION — BREAK DOWN DATA AND PROBLEMS INTO SMALLER PARTS

PATTERN RECOGNITION — OBSERVE PATTERNS AND TRENDS IN DATA

ALGORITHMS — DETERMINE WHAT STEPS ARE NEEDED TO SOLVE A PROBLEM

ABSTRACTION — REMOVE DETAILS AND EXTRACT RELEVANT INFORMATION

Programming Guidelines

- **Structure & Design:**
  - Breaking the problem into manageable parts.
  - Using functions/classes for modularity.
  - Commenting the code for readability.
- **Algorithm design:**
  - Choosing efficient algorithms for the problem.
  - Considering stability, convergence, and computational cost.
- **Programming skills:**
  - Implementing algorithms in languages like Python, C++, or Fortran.
  - Utilizing libraries and numerical frameworks.
- **Debugging and optimization:**
  - Identifying and fixing errors for accurate results.
  - Improving efficiency and scalability.

"Think about the problem, not the Program."

Testing the Code

- **Verification:**
  - Checking if the code solves the intended problem.
  - Comparing outputs to analytical solutions or benchmark cases.
- **Validation:**
  - Comparing results to experimental data or real-world observations.
  - Confirming the code produces realistic and meaningful output.
- **Sensitivity analysis:**
  - Assessing how results change with parameter variations.
  - Understanding model robustness and limitations.

Assessing Errors

- **Truncation errors:**
  - Approximations inherent in numerical methods.
  - Decrease with finer resolution (e.g., grid size).
- **Round-off errors:**
  - Finite precision of floating-point arithmetic.
  - Accumulate over calculations.
- **Convergence analysis:**
  - Studying how errors change with computational resources.
  - Insights into accuracy and stability.

Languages in Computational Physics: Features Summary

Python

- Clear syntax, easy learning.
- Versatile for diverse tasks.
- Rich libraries: NumPy, SciPy.

C++

- High performance, memory efficiency.
- Strongly typed, object-oriented.
- Extensive libraries: Eigen, Boost.

Fortran

- Historical significance in computing.
- High-performance numerical operations.
- Specialized scientific functions.

Why Python?

- **Versatility:** Handles diverse tasks beyond numerical computation, including data manipulation, visualization, and machine learning.
- **Rich Libraries:** Comprehensive scientific ecosystem (NumPy, SciPy, Matplotlib, etc.) covering mathematics, physics, signal processing, and statistical analysis.
- **Rapid Prototyping:** Faster development cycle due to its clear syntax and interpreted nature, enabling quick testing of algorithms and ideas.
- **Interactivity:** Supports interactive data exploration and visualization, making it ideal for scientific research and analysis.
- **Readability:** Clear and intuitive syntax aids collaboration among researchers and scientists, facilitating code maintenance and sharing.
- **Integration:** Seamlessly integrates with performance-oriented languages like C/C++/Fortran for computationally intensive tasks, combining ease of development with optimized performance.

Python basics for getting started.

- Structure of Python Program.
- Basic I/O, Conditional statements.
- Functions and Python Data Types.
- Array operations with NumPy.
- Graphics with Matplotlib.
- File handling
- Code optimization

Introduction to Python Variables

- **Variables**: In Python, variables are used to store data values.
- **Declaration**: Unlike some other programming languages, Python doesn't require explicit declaration of variables.
- **Assigning Values**: You can assign values to variables using the assignment operator =.
- **Variable Naming Rules**:
  - Cannot start with a digit.
  - Can contain letters, numbers, and underscores.
  - Case-sensitive (`myVar` and `myvar` are different).
  - Python has reserved keywords like if, else, for, while, etc., which cannot be used as variable names.
- **Examples**:
  - `num = 10`
  - `name = "Alice"`
  - `is_valid = True`

Python Data Types

- **Numeric Types**:
  - `int`: Integers (e.g., 5, -7, 0)
  - `float`: Floating-point numbers (e.g., 3.14, -0.5)
- **Sequence Types**:
  - `str`: Strings (e.g., 'hello', "Python")
  - `list`: Lists (e.g., [1, 2, 3], ['a', 'b', 'c'])
  - `tuple`: Tuples (e.g., (1, 2, 3), ('a', 'b', 'c'))
- **Boolean Type**:
  - `bool`: Boolean values (True or False)
- **Mapping Type**:
  - `dict`: Dictionaries (e.g., 'name': 'Alice', 'age': 30)
- **Set Type**:
  - `set`: Unordered collection of unique elements (e.g., 1, 2, 3)
- **None Type**:
  - `None`: Represents absence of a value or a null value

Lists, Tuples, Dictionaries, and Indexing

Lists

- Ordered, mutable collection.
- Example: `my_list = [1, 2, 3, ‘hello’]`
- Indexing: `my_list[2]` yields 3.

Tuples

- Ordered, immutable collection.
- Example: `my_tuple = (1, 2, ‘world’)`
- Indexing: Similar to lists.

Dictionaries

- Unordered, key-value pairs.
- Example: `my_dict = {’name’: ‘Alice’, ‘age’: 30}`
- Indexing: `my_dict[‘name’]` yields `‘Alice’`.

Python Control Loops

1. `for` Loop

- Iterates over a sequence (list, tuple, string, etc.).
- Syntax:

```
for item in sequence:
    # code block
```

- Example:

```
numbers = [1, 2, 3, 4, 5]
for num in numbers:
    print(num)
```

Python Control Loops

2. `while` Loop

- Executes a block of code as long as the specified condition is true.
- Syntax:

```
while condition:
    # code block
```

- Example:

```
count = 0
while count < 5:
    print(count)
    count += 1
```

NumPy and Matplotlib Overview

- **NumPy**:
  - Powerful Python library for numerical operations.
  - Provides high-performance multidimensional arrays.
  - Facilitates mathematical, logical, and statistical operations.
  - Essential for scientific computing and data analysis.
- **Matplotlib**:
  - Main Python plotting library for 2D graphics.
  - Creates versatile plots: line, scatter, bar, etc.
  - Highly customizable with labels, legends, colors, and more.
  - Widely used for visualizing data and scientific graphics.

Saving and Loading NumPy Data

- **Saving NumPy Data**:
    - Use `np.save()` to save NumPy arrays to a file.
    - Example: `np.save('saved_data.npy', data_to_save)`.
- **Loading NumPy Data**:
    - Use `np.load()` to load saved NumPy arrays from a file.
    - Example: `loaded_data = np.load('saved_data.npy')`.

Radioactive Decay

Radioactive Decay

- $^{235}U \rightarrow 143$ neutrons $+$ 92 protons.
- Random $\rightarrow$ not be able to predict precisely.
- Avg. time for decay 10^9 years (mean life time).
- Let's consider a 235^U sample. Say, $N_U(t)$ is the no. of uranium nuclei at time t.
$$\frac{dN_U}{dt} = -\frac{N_U}{\tau}, \tag{4}$$
where τ is the time constant for decay.
- $$N_U(t) = N_U(0)e^{-t/\tau}$$
.

Numerical Approach

- Useful to introduce several computational methods.
- Now, let's start with a simple method.
- Goal $\rightarrow$ To obtain $N_U(t) \rightarrow$ IVP.
- Taylor expansion for N_U at $t = 0$

$$N_U(\Delta t) = N_U(0) + \frac{dN_U}{dt}\Delta t + \frac{1}{2}\frac{d^2 N_U}{dt^2}(\Delta t)^2 + \dots \tag{5}$$

- for $\Delta t << 1$,

$$N_U(\Delta t) \approx N_U(0) + \frac{dN_U}{dt}\Delta t \tag{6}$$

Derivative of N_U

- $$\frac{dN_U}{dt}\Delta t = \lim_{\Delta t \to 0} \frac{N_U(\Delta t + t) - N_U(t)}{\Delta t} \approx \lim_{\Delta t \to 0} \frac{N_U(\Delta t + t) - N_U(t)}{\Delta t}$$
- From the physics of the problem, we know $\frac{dN_U}{dt} = -\frac{N_U}{\tau}$

$$\implies N_U(t + \Delta t) \approx N_U(t) - \frac{N_U(t)}{\tau}\Delta t \tag{7}$$

- Numerical sol. of the radioactive decay problem.
- $t = \Delta t, 2\Delta, \dots, n\Delta t \implies N_U(n\Delta t)$
- only an approximation $\implies$ minimize the error.
- $\implies$ Euler method, a useful general algorithm to solve ODEs.

Construction of a working program

- Describe the structure of a program in a general way - pseudocode.
- Translate each piece of the pseudocode into specific instructions.
- Highly individualized process.
- The First thing to do is to think. "Think about the problem, not the program".
- Construct an outline – How to solve.
- Variables/Parameters available.
- For the radioactive decay problem, $N_U, t, \tau, \Delta t$ and goal is to calculate $N_U(t)$.
- We will use an array to store N_U.
- General plan is to apply Eq. (28) repetitively.

Pseudocode

- Comment text.
- Declare necessary variables and arrays.
- Initialize the variables.
- Do the actual calculations.
- Store the results.

Testing your program

- It is not a working program until the output is correct!
- Checking the program is not always a trivial task.
- Does the output look reasonable?
- Does the output agree with any exact results that are available?
- Check for different Δt.

Numerical Considerations

- How to estimate numerical errors associated with
- technique/algorithm.
- In the radioactive decay, approximation.
- Finite numerical precision (round-off errors).
- Use double precision.
- No guarantee that these errors will be negligible.
- It is difficult to give any general rule about what to watch for?

Ex: Radioactive Decay

- ODE $\rightarrow$ difference equation.
- $N_U(n\Delta t)$
- Discretization in time or space, or both.
- The errors introduced by the discreteness are negligible?
- How to choose Δt ?
- No general answer $\rightarrow$ can only give some guidelines.
- Always repeat using different Δt.
- More quantitatively, $(\Delta t)^2, \; t/\Delta t$
- Total (global) error $\propto (t/\Delta t) \times (\Delta t)^2 = \Delta t$
- Δt ¡ time scale.
- A very reasonable numerical approach can be inherently unstable.

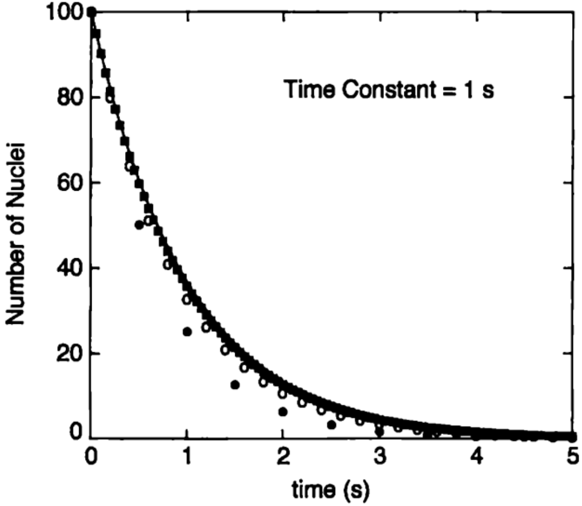


Figure: Numerical solution of radioactive decay problem.

Numerical Methods for Solving the Differential Equations

Coupled ODE's

- Often, most systems & their dynamics are represented by Coupled Differential equations with more than one variable.

$$\begin{aligned}
\frac{dx_1(t)}{dt} &= f_1(x_1, x_2, ..., x_n, t)\\
\frac{dx_2(t)}{dt} &= f_2(x_1, x_2, ..., x_n, t)\\
&\vdots\\
\frac{dx_n(t)}{dt} &= f_n(x_1, x_2, ..., x_n, t)
\end{aligned} \tag{8}$$

- In compact notation

$$\frac{d\boldsymbol{X}(t)}{dt} = \boldsymbol{F}(\boldsymbol{X}(t), t) \tag{9}$$

where $\boldsymbol{X} = (x_1, x_2, ..., x_n)^T$ and $\boldsymbol{F} = (f_1, f_2, ..., f_n)^T$

For instance,

$$\dot{x} = \sigma(y - x) \tag{10}$$
$$\dot{y} = rx - y - xz \tag{11}$$
$$\dot{z} = -bz + xy \tag{12}$$

The equations describe the rate of change of three quantities with respect to time:

- x is proportional to the rate of convection,
- y to the horizontal temperature variation,
- z to the vertical temperature variation.

The constants σ, r, and b are system parameters proportional to the Prandtl number, Rayleigh number, and certain physical dimensions of the layer itself.

Euler's Method for N-Coupled ODEs

- $$\boldsymbol{X}(\Delta t) = \boldsymbol{X}(0) + \Delta t \left.\frac{d\boldsymbol{X}}{dt}\right|_{t=0} + \frac{\Delta t^2}{2}\left.\frac{d^2\boldsymbol{X}}{dt^2}\right|_{t=0} + \dots \tag{13}$$

- $$\implies \boldsymbol{X}(\Delta t) \approx \boldsymbol{X}(0) + \left.\frac{d\boldsymbol{X}}{dt}\right|_{t=0} \Delta t \tag{14}$$

- $$\implies \boldsymbol{X}(t+h) = \boldsymbol{X}(t) + h\boldsymbol{F}(\boldsymbol{X}(t), t) \tag{15}$$

  $\implies$ Euler's Method

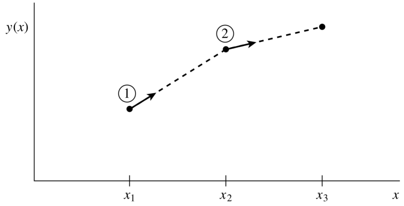


Figure: Euler's Method for integrating an ODE.

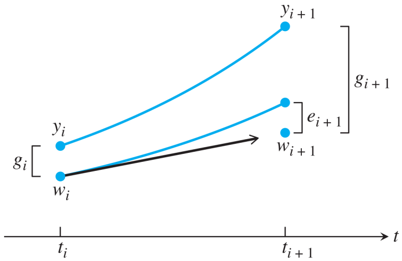


Figure: One step of an ODE solver.

Runge-Kutta 2th Order (RK2)

- Consider the IVP

$$\frac{dy(t)}{dt} = y'(t) = f(y(t), t) \tag{16}$$

with $y(t_0) = y_0 \rightarrow$ initial condition

- Now, let us define two approximation to the derivative as

$$k_1 = f(y(t_0), t_0) \tag{17}$$
$$k_2 = f(y(t_0) + \beta h k_1, t_0 + \alpha h) \tag{18}$$

- In all cases α & β will represent fractional values btw. 0 and 1.

- Eq. (17) $\implies$ k_1 is the approximation to the derivative based on the estimated value of $y(t)$ at $t = t_0$ i.e. y_0.
- Eq. (18) $\implies$ k_2 is the approximation to the derivative based on the estimate value $y(t_0) + \beta h \times$ slope k_1, at $t = t_0 + \alpha h$
- To update the solution with the next estimate of y(t) at $t_0 + h$, we use

$$y(t_0 + h) = y(t_0) + h(ak_1 + bk_2) \tag{19}$$

- Note that Euler's Method (1^{st} order RK) is a special case of this method with a = 1 & b = 0.
- Now our task is to determine, how to find the values of α, β, a & b that result in low error.

- Eq. (19) $\implies$

$$y(t_0 + h) = y(t_0) + h(af(y,t) + bf(y + \beta hk_1, t_0 + \alpha h)) \tag{20}$$

using Taylor's series expansion for the rightmost term.

$$\begin{aligned} f(y + \beta hk_1, t_0 + \alpha h) &= f(y, t_0) + f_y \beta hk_1 + f_t \alpha h + \ldots \\ &= f + f_y f \beta h + f_t \alpha h + \ldots \end{aligned} \tag{21}$$

- Eq. (20) $\implies$

$$\begin{aligned} y(t_0 + h) &= y(t_0) + h\left[af + b(f + f_y f \beta h + f_t \alpha h + \ldots)\right] \\ &= y(t_0) + haf + hbf + h^2 \beta bf_y f + h^2 f_t \alpha b + \ldots \\ &= y(t_0) + h(a+b)f + h^2 f_t \alpha b + h^2 \beta bf_y f + \ldots \end{aligned} \tag{22}$$

- Now, we compare this approximation with the expression for a Taylor expansion of the exact solution.

$$\begin{aligned} y(t_0+h) &= y(t_0) + hy'(t)\Big|_{t=t_0} + \frac{h^2}{2}y''(t)\Big|_{t_0} + \dots \\ &= y(t_0) + hf + \frac{h^2}{2}(f_t + f_y f) + \dots \\ &= y(t_0) + hf + \frac{h^2}{2}f_t + \frac{h^2}{2}f_y f + \dots \end{aligned} \tag{23}$$

- Comparing Eq. (22) & Eq. (23),

$$\begin{aligned} a+b &= 1 \\ \alpha b &= \frac{1}{2} \\ \beta b &= \frac{1}{2} \end{aligned} \tag{24}$$

- This system is underspecified, there are 4 unknowns, and only 3 equations. Hence more than one solution is possible. $\therefore$ Clearly, $a=0,\ b=1\ \&\ \alpha=\beta=\frac{1}{2}$ is a possible sol.

- Since all of the terms of the approximation are equal to the terms in the exact solution, upto the error terms, the local error of this method is therefore $O(h^3)$.
- Another common choice for the coefficients are

$$a = b = \frac{1}{2}\ ,\ \alpha = \beta = 1.$$

- Now the Second order Runge-Kutte Method is

$$\begin{aligned} k_1 &= f(y(t_0), t_0) \\ k_2 &= f(y(t_0) + hk_1, t_0 + h) \\ y(t_0 + h) &= y(t_0) + h\left(\frac{k_1 + k_2}{2}\right) \end{aligned} \tag{25}$$

- In terms of algorithm description

$$k_1 = f(y(t_0), t_0) \implies \text{estimation of derivative at t} = t_0$$

$$y_1(t_0 + h) = y(t_0) + hk_1 \implies \text{inter. estimate of func. at t} = t_0 + \text{h}$$

$$k_2 = f(y_1(t_0 + h), t_0 + h) \implies \text{estimte of slope at t} = t_0 + \text{h}$$

$$y(t_0 + h) = y(t_0) + h\left(\frac{k_1 + k_2}{2}\right) \implies \text{estimate of } y(t_0 + h) \text{ using}$$

average of slopes

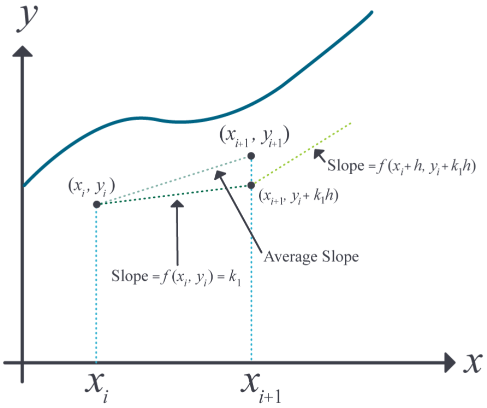


Figure: Runge-Kutta 2nd order method (Heun's method).

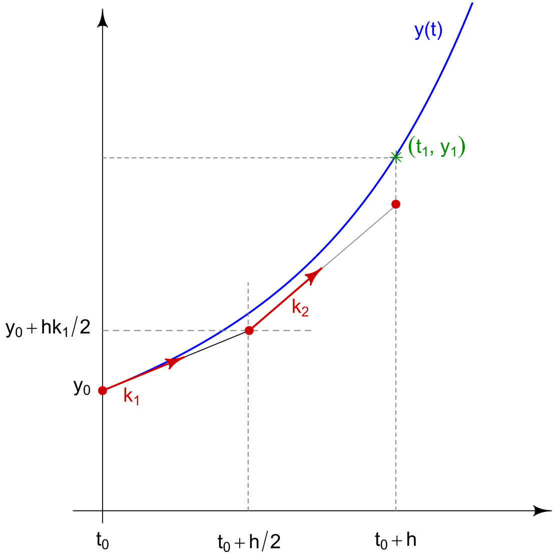


Figure: Slopes used by Runge-Kutta (RK2) Method

Validation

- $$\frac{dy}{dt} = -2y, y(0) = 3$$
- Exact sol. $y(t) = 3e^{-2t}$
  Choose h = 0.2, so that t = t_0 + h = 0.2 as t_0 = 0.

$$\begin{aligned} k_1 &= f(y(t_0), t_0) \\ &= 3e^{-2t_0} = -6 \end{aligned} \tag{26}$$

$$k_2 = f(y_1(t_0) + hk_1, t_0 + h) = -4.0219 \tag{27}$$

$$\begin{aligned} y(h = 0.2) &= y(0) + h\left(\frac{k_1 + k_2}{2}\right) \\ &= 3 + 0.2(-5.0110) = 1.9978 \end{aligned} \tag{28}$$

The exact sol. at t = h = 0.2 is y(0.2) = 2.0110

- Check by plotting exact sol. & numerically for h = 0.8, 0.2, 0.01

Second Order Runge-Kutta

- $$\frac{dy(t)}{dt} = f(y(t), t)$$
- Ansatz to the solution

$$y(t+h) = y(t) + h(ak_1 + bk_2),$$

where

$$k_1 = f(y(t), t), \tag{29}$$
$$k_2 = f(y(t) + \beta h k_1, t + \alpha h), \tag{30}$$

Here a, b, α, β $\rightarrow$ unknown constants.

- Taylor's Series Expansion of the solution upto 2^{nd} order in 'h'.

$$\implies y(t+h) = y(t_0) + h(a+b)f + h^2 f_t \alpha b + h^2 \beta b f_y f + \ldots \tag{31}$$

- Comparing $\implies$ a + b = 1, $\alpha b = \frac{1}{2}$, $\beta b = \frac{1}{2}$

$$(a, b, \alpha, \beta) = \left(0, 1, \frac{1}{2}, \frac{1}{2}\right) \quad \& \quad \left(\frac{1}{2}, \frac{1}{2}, 1, 1\right)$$

- Solution to the differential equation

$$y(t+h) = y(t) + \frac{h}{2}(k_1 + k_2) \tag{32}$$

Euler vs RK2 (Computational Cost)

- Nuclear Decay: $N_U(\Delta t) \approx N_U(0) - (\Delta t/\tau)N_U(0)$
- If $\Delta t = \tau \implies N_U(t) \approx 0$
- If $\Delta t > \tau \implies N_U(t) \to -, +, -, +, \dots$ (even worse).
- For Euler: $f(y,t) \to$ accurate upto Δt.
- For RK2: $k_1 = f(y,t)$ & $k_2 = f(y + \beta h k_1, t + \alpha h)$
  $\to$ accurate upto $(\Delta t)^2$.
- Cost to increase in accuracy $\to$ twice computing requirement of Euler method.
- For $\Delta t = 0.01$, RK2 is 100 times accurate than Euler.
- To achieve same level of accuracy using Euler method, $\Delta t =?$

Simple Harmonic Motion

- The equation of motion for this problem is

$$\frac{d^2y}{dx^2} = -Ay \tag{33}$$

where A is a constant.

- The total energy of the oscillator is

$$E = \frac{1}{2}v^2 + \frac{1}{2}Ay^2 \tag{34}$$

Here $v = dy/dt$, first term is kinetic energy (mass is taken as unity), and second term is potential energy.

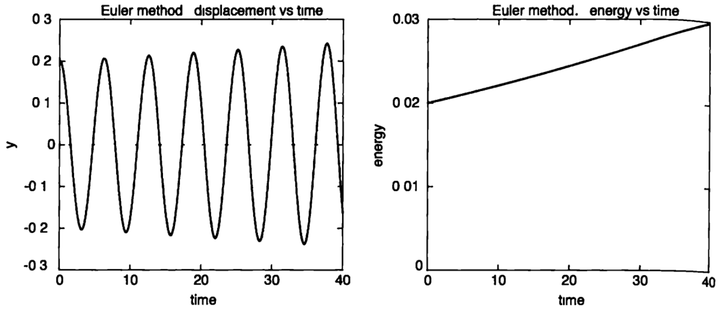


Figure: Simulation of simple harmonic oscillator with Euler Method.

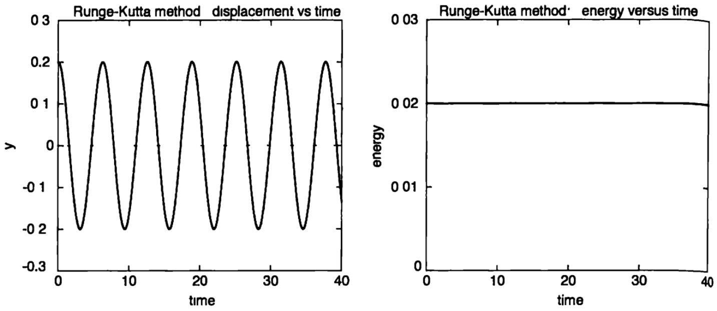


Figure: Simulation of simple harmonic oscillator with Runge-Kutta Method.

Third-Order Runge-Kutta

- Ansatz to the solution

$$y(t+h) = y(t) + h\left(ak_1 + bk_2 + ck_3\right)$$

where

$$\begin{aligned} k_1 &= f(y(t), t), \\ k_2 &= f(y(t) + \beta_1 h k_1, t + \alpha_1 h), \\ k_3 &= f(y(t) + h(\beta_2 k_1 + \beta_3 k_2), t + \alpha_2 h). \end{aligned} \tag{35}$$

- 8 unknown parameters $(a, b, c, \alpha_1, \alpha_2, \beta_1, \beta_2, \beta_3)$.
- Comparing Taylor Series expansion of the solution upto 3^{rd} order in 'h'.

$$\Longrightarrow \quad \alpha_1 = \beta_1 \quad ; \quad \alpha_2 = \beta_2 + \beta_3 \quad ; \quad a + b + c = 1 \quad ; \tag{36}$$

$$b\alpha_1 + c\alpha_2 = \frac{1}{2} \quad ; \quad b\alpha_1^2 + c\alpha_2^2 = \frac{1}{3} \quad ; \quad c\alpha_1\beta_3 = \frac{1}{6} \tag{37}$$

- If we choose $\alpha_1 = \frac{1}{2}$; $\alpha_2 = 1$
- Values of the other parameters will be $\beta_1 = \frac{1}{2}, \beta_2 = -1, \beta_3 = 2, a = 1/6, b = \frac{2}{3}, c = \frac{1}{6}$
- Solution to the differential equation

$$y(t+h) = y(t) + \frac{h}{6}\left(k_1 + 4k_2 + k_3\right)$$

where,

$$\begin{aligned} k_1 &= f(y(t), t) \\ k_2 &= f\left(y(t) + \frac{h}{2}k_1, t + \frac{h}{2}\right) \\ k_3 &= f(y(t) + h\left(2k_2 - k_1\right), t + h) \end{aligned} \tag{38}$$

Fourth-Order Runge-Kutta

- Ansatz to the solution

$$y(t+h) = y(t) + h(ak_1 + bk_2 + ck_3 + dk_4)$$

where,

$$\begin{aligned} k_1 &= f(y(t), t) \\ k_2 &= f(y(t) + \beta_1 hk_1, t + \alpha_1 h) \\ k_3 &= f(y(t) + h(\beta_2 k_1 + \beta_3 k_2), t + \alpha_2 h) \\ k_4 &= f(y(t) + h(\beta_4 k_1 + \beta_5 k_2 + \beta_6 k_3), t + \alpha_3 h) \end{aligned} \tag{39}$$

here we have 13 unknown parameters.

- Comparing Taylor's Series expansion of the solution upto 4th order in 'h' yields 10 independent equations.

$$\alpha_1 = \beta_1; \quad \alpha_2 = \beta_2 + \beta_3; \quad \alpha_3 = \beta_4 + \beta_5 + \beta_6 \quad ;$$

$$a + b + c + d = 1; \quad b\alpha_2 + c\alpha_3 + d\alpha_4 = \frac{1}{2}; \quad b\alpha_1^2 + c\alpha_2^2 + d\alpha_3^2 = \frac{1}{3}$$

$$b\alpha_1^3 + c\alpha_2^2 + d\alpha_3^2 = \frac{1}{4} \quad ; \quad c\alpha_1\beta_3 + d(\alpha_1\beta_5 + \alpha_2\beta_6) = \frac{1}{6}$$

$$c\alpha_1\alpha_2\beta_3 + d\alpha_3(\alpha_1\beta_5 + \alpha_2\beta_6) = \frac{1}{8} \quad ; \quad d\alpha_1\beta_3\beta_6 = \frac{1}{24}$$

$$c\alpha_1^2\beta_3 + d(\alpha_1^2\beta_5 + \alpha_2^2\beta_6) = \frac{1}{12}$$

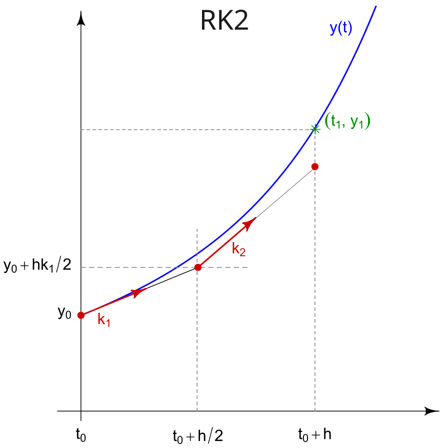
- If we choose $\alpha_1 = \frac{1}{2}$, $\beta_2 = 0$
- $$\alpha_2 = \frac{1}{2} \quad ; \quad \alpha_3 = 1 \quad ; \quad \beta_1 = \beta_3 = \frac{1}{2} \quad ; \quad \beta_4 = \beta_5 = 0 \quad ;$$

$$\beta_6 = 1 \quad ; \quad a = \frac{1}{6} \quad ; \quad b = c = \frac{1}{3} \quad ; \quad d = \frac{1}{6}$$

4th order Runge-Kutta

$$y(t_0 + h) = y(t_0) + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

$$\begin{aligned}
k_1 &= f(y_0, t_0), \\
k_2 &= f(y_0 + \frac{hk_1}{2}, t_0 + \frac{h}{2}), \\
k_3 &= f(y_0 + \frac{hk_2}{2}, t_0 + \frac{h}{2}), \\
k_4 &= f(y_0 + hk_3, t_0 + h).
\end{aligned} \tag{40}$$



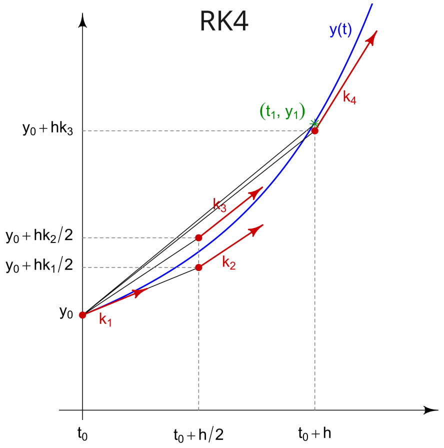


Figure: Slopes used by Runge-Kutta Methods

Centered Difference Method: Verlet Method

- Euler, RK methods $\rightarrow$ extrapolation in asymmetric manner.
- Involves only "forward" looking in the estimation of derivatives.
- Simple but more "symmertric approaches" $\rightarrow$ smaller numerical errors.
- Basis of centered difference methods.

Verlet Method

- $$\frac{d^2y}{dt^2} = f_2(y,t) \implies \frac{dy}{dt} = v; \ \frac{dv}{dt} = f_2(y,t)$$
- Euler's method:

$$\begin{aligned} y(t_0+h) &\approx y(t_0) + v(t_0)\Delta t \\ v(t_0+h) &\approx v(t_0) + f_2(y_0(t_0), t_0)\Delta t \end{aligned}$$

- Interested only in $y(t)$ if f_2 does not depend on v,
  $\implies$ we can skip dealing with the first derivatives entirely.

- $$y(t_0+h) = y(t_0)+\frac{dy}{dt}\bigg|_{t=t0}\Delta t+\frac{1}{2}\frac{d^2y}{dt^2}\bigg|_{t=t0}(\Delta t)^2+\frac{1}{6}\frac{d^3y}{dt^3}\bigg|_{t=t0}(\Delta t)^3+\ldots$$
  $$y(t_0-h) = y(t_0)-\frac{dy}{dt}\bigg|_{t=t0}\Delta t+\frac{1}{2}\frac{d^2y}{dt^2}\bigg|_{t=t0}(\Delta t)^2-\frac{1}{6}\frac{d^3y}{dt^3}\bigg|_{t=t0}(\Delta t)^3+\ldots$$
- $y(t_0+h) \approx 2y(t_0) - y(t_0+h) + \frac{d^2y}{dt^2}(\Delta t)^2 + O([\Delta t]^4)$
- We define the approximate solution for y as

$$y(t_0+h) \approx 2y(t_0) - y(t_0-h) + f_2(y(t_0), t_0)(\Delta t)^2$$

- Lowest-order correlation term is of $O([\Delta t]^4)$
  $\implies$ Verlet Method
- $$v(t_0) \approx \frac{y(t_0+h) - y(t_0-h)}{2\Delta t}$$
- Error is of $O([\Delta t]^2) \implies$ Larger than that of 'y'

Trajectory of a cannonball with varying air drag

- Consider a cannonball shot from the ground.
- Motion of cannonball is restricted to two dimension (XY Plane).

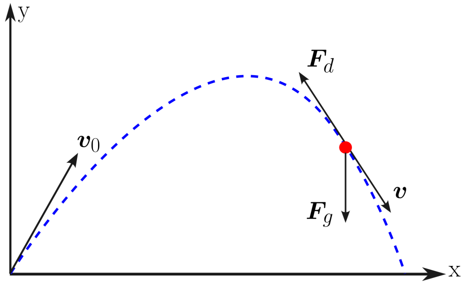


Figure: Trajectory of cannonball with air drag.

- The net force acting on cannonball is

$$\boldsymbol{F} = \boldsymbol{F}_g + \boldsymbol{F}_d = -mg\,\hat{\boldsymbol{y}} - \alpha|\dot{\boldsymbol{r}}|^2 \frac{\dot{\boldsymbol{r}}}{|\dot{\boldsymbol{r}}|}$$

  where $\boldsymbol{r}$ and $\dot{\boldsymbol{r}}$ are respectively the position and velocity, m is the mass of the cannonball, g is the acceleration due to gravity, and $\alpha = \frac{1}{2}\rho C_d A$. C_d is the drag coefficient, A is the cross-sectional area of the projectile, and ρ is the density of the air.

- According to the Newton's 2nd law of motion,

$$m\frac{d^2\mathbf{r}}{dt^2} = \mathbf{F} = -mg\hat{\mathbf{y}} - \frac{\rho C_d A}{2}\left[\frac{dx}{dt}\left\{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2\right\}^{1/2}\hat{\mathbf{x}} + \frac{dy}{dt}\left\{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2\right\}^{1/2}\hat{\mathbf{y}}\right]$$

- Equating the x and y components of the above equation.

$$\frac{d^2x}{dt^2} = -\frac{\pi R^2\rho C_d}{2m}\frac{dx}{dt}\left\{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2\right\}^{1/2}$$

$$\frac{d^2y}{dt^2} = -g - \frac{\pi R^2\rho C_d}{2m}\frac{dy}{dt}\left\{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2\right\}^{1/2}$$

- In the absence of air-resistance, motion of the cannonball is independent of its mass. But in real-world mass certainly makes difference.

- We can rewrite the two 2nd order ODEs into four first order coupled ODEs.

$$
\begin{aligned}
\frac{dx}{dt} &= v_x \\
\frac{dv_x}{dt} &= -\frac{\pi R^2 \rho C_d}{2m} v_x (v_x^2 + v_y^2)^{1/2} \\
\frac{dy}{dt} &= v_y \\
\frac{dv_y}{dt} &= -g - \frac{\pi R^2 \rho C_d}{2m} v_y (v_x^2 + v_y^2)^{1/2}
\end{aligned}
$$

- To determine the trajectory, we need to provide initial conditions.

$$
x\Big|_{t=0} = x_0 \qquad\qquad y\Big|_{t=0} = y_0
$$

$$
v_x\Big|_{t=0} = v_{x_0} \qquad\qquad v_y\Big|_{t=0} = v_{y_0}
$$

Chaos in Nonlinear Oscillator

Linear harmonic oscillator

$$\frac{d^2y}{dt^2} + \omega_0^2 y = 0$$

- $y(t) = A\cos\omega_0 t$; IC's $y(0) = A$; $\dot{y}(0) = 0$
- SHO: $T = 2\pi/\omega$; independent of amplitude.

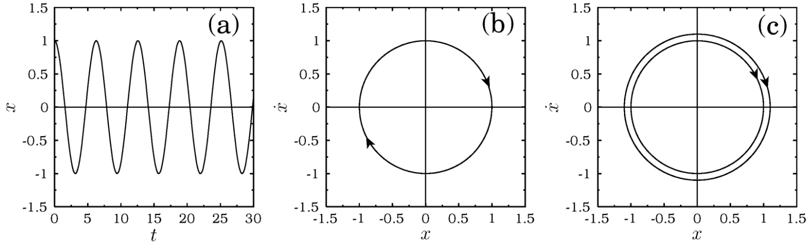


Figure: (a) Solution curve and, (b) phase portrait of the free linear harmonic oscillator equation, (c) depicts the nearby periodic orbits with initial conditions (1,0) and (1.1,0).

Linear damped oscillator

$$\frac{d^2y}{dt^2} + \alpha\frac{dy}{dt} + \omega_0^2 y = 0$$

- $y(t) = A_1 \exp(m_1 t) + A_2 \exp(m_2 t)$; IC's $y(0) = A$; $\dot{y}(0) = 0$
- $$m_{1,2} = \frac{1}{2}\left[-\alpha \pm \sqrt{\alpha^2 - 4\omega_0^2}\right]$$
- Under damping: $0 < \alpha < 2\omega_0$

$$y(t) = \frac{A\omega_0}{c}\exp(-\alpha t/2)\cos(ct - \delta),$$

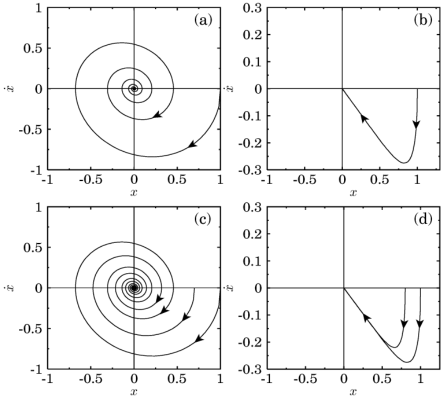
where $c = \sqrt{\omega_0^2 - \alpha^2/4}$, $\delta = \tan^{-1}\alpha/2c$

- Critical damping: $\alpha = 2\omega_0$

$$y(t) = A\exp(-\alpha t/2)\,(1 + \alpha t/2)\,,$$

where $y(0) = A$; $\dot{y}(0) = 0$

- Over damping: $\alpha > 2\omega_0$
- y(t) = ? ; $y(0) = A$; $\dot{y}(0) = 0$



Linear damped and forced oscillator

- $$\frac{d^2y}{dt^2} + \alpha\frac{dy}{dt} + \omega_0^2 y = f \sin\omega t$$
- $$y(t) = \frac{A_t\omega_0}{c} e^{-\alpha t/2}\cos(ct - \delta) + A_p\cos(\omega t - \gamma)$$
- $$A_p = \frac{f}{(\omega_0^2 - \omega^2) + \alpha^2\omega^2}; \quad \gamma = \tan^{-1}\left(\frac{\omega^2 - \omega_0^2}{\alpha\omega}\right)$$
- $T = 2\pi/\omega$, A_p is its amplitude.
- Resonance $\omega = \sqrt{\omega_0^2 - (\alpha^2/2)} \quad (\approx \omega_0,\ \alpha << 1)$
- $A_{p(max)} =?$

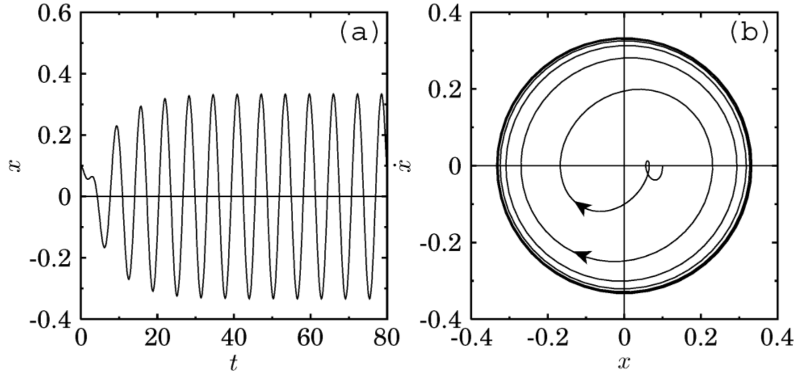


Figure: (a) Solution curve and (b) phase trajectory of the system for $\alpha = 0.1$, $\omega_0^2 = 1$, $\omega = 1$ and $f = 0.1$

Damped and driven nonlinear oscillator

- $$\frac{d^2y}{dt^2} + \alpha\frac{dy}{dt} + \beta y^3 + \omega_0^2 y = f \sin \omega t$$
  $$\implies \text{Duffing Oscillator}$$

- $$V(x) = \frac{1}{2}\omega_0^2 x^2 + \frac{\beta}{4}x^4$$

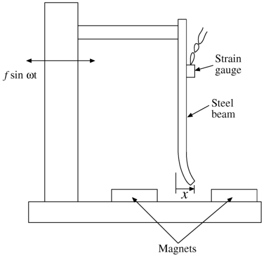


Figure: A mechanical model of the Duffing equation.

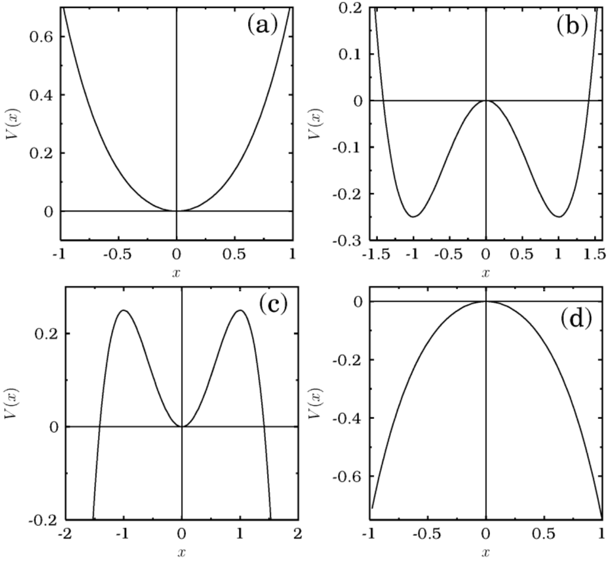


Figure: Shape of the potential function.

Free oscillation($\alpha = \beta = 0$)

- $\omega_0^2 > 0,\ \beta > 0,\ y(0) = A,\ \dot{y}(0) = 0$
- $y(t) = Acn(\bar{\omega}t; k)$, $\bar{\omega} = \sqrt{\omega_0^2 + \beta A^2}$

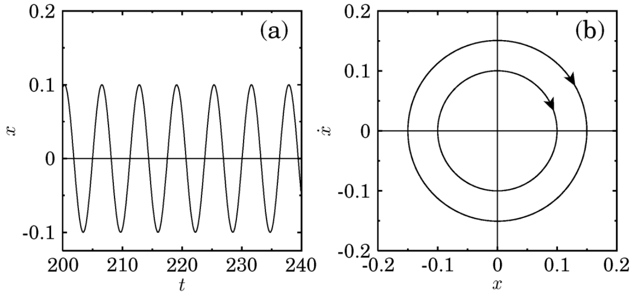


Figure: Trajectory Plot.

Damped oscillations

- underdamped case $\alpha < 2\omega_0$
- $\alpha = \pm(3/\sqrt{2})\omega_0$, & $\beta > 0$
- $y(t) = (\omega_0/\sqrt{\beta}) A \exp\left(-\omega_0 t/\sqrt{2}\right) cn(Av; k)$,
- $v = -\sqrt{2}\exp\left(-\omega_0 t/\sqrt{2}\right) - v_0$

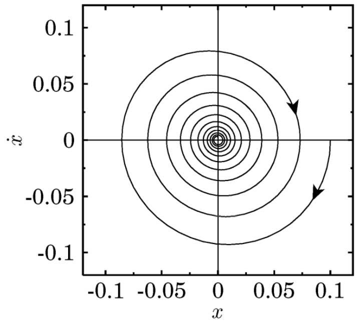


Figure: Damped oscillatory solution of the nonlinear equation.

Forced Oscillations: Primary Resonance and Hysteresis

- Assume $y(t) = A\sin(\omega t + \delta)$:$0 < \beta << 1$ & $\omega \approx \omega_0$ (Primary Resonance)
- $[(\omega_0^2 - \omega^2)A + (3/4)\beta A^3]^2 + (\alpha A\omega)^2 = f^2$
- $\implies$ frequency-response equation (curve).
- Leans to the right of $\omega = \omega_0$ for $\beta > 0$
- Near $\omega = \omega_0$, we have the resonances.

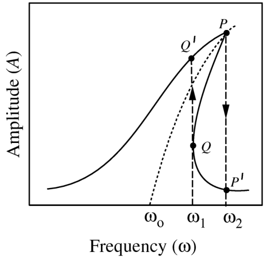


Figure: A typical frequency-response curve of the nonlinear equation for $\beta > 0$.

Period-doubling route to chaos

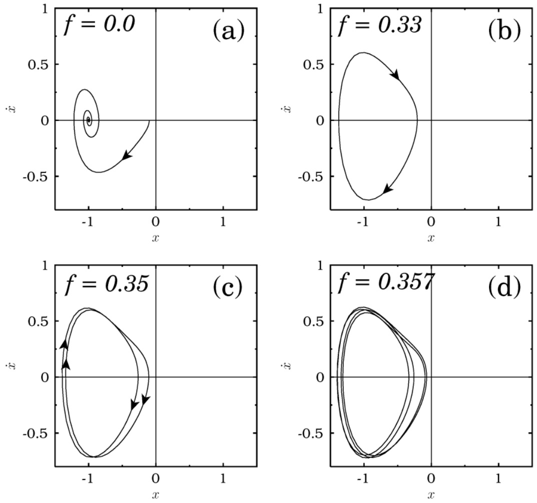


Figure: Phase portraits of the Duffing oscillator equation.

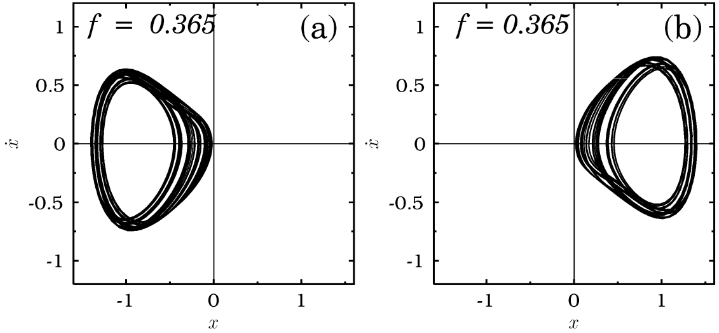


Figure: One-band chaotic attractors confined to (a) the left well and (b) the right well for $f = 0.365$

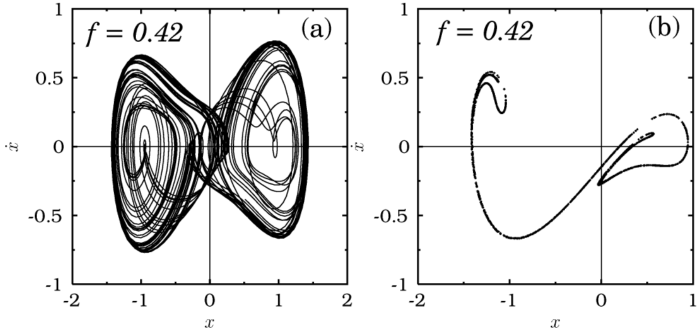
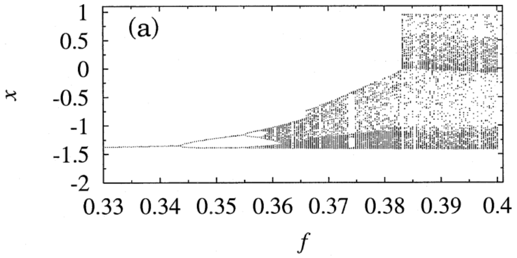


Figure: (a) Phase portrait and (b) Poincare map of the double-band chaotic attractor for $f = 0.42$



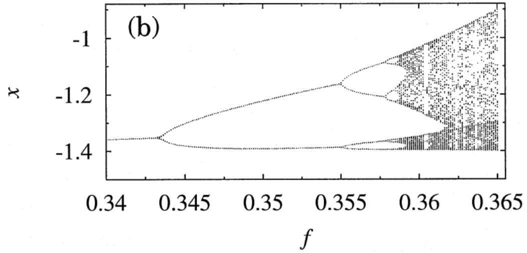


Figure: (a,b) Bifurcation sequence of the left well attractor.

Value of f	Nature of solution
$f = 0$	Damped oscillation to the stable spiral equilibrium $x_1^* = -1, y_1^* = 0$
$0 < f < 0.3437$	Period-T oscillation
$0.3437 \leq f < 0.355$	Period-$2T$ oscillation
$0.355 \leq f < 0.3577$	Period-$4T$ oscillation
$0.3577 \leq f < 0.3589$	Further period doublings
$0.3589 \leq f < 0.3833$	One-band chaos
$0.3833 \leq f \leq 0.42$	Double-band chaos
$f > 0.42$	Chaos, periodic windows, period doublings of windows, reverse period doublings, etc.

Problems

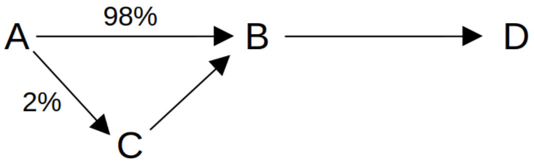
Problem 1

(a) Consider an example where the decay chain involves four isotopes P, Q, R, and S such that

$$\begin{tikzcd} P \arrow[r, "3.11 \text{ min}"] & Q \arrow[r, "26.8 \text{ min}"] & R \arrow[r, "19.9 \text{ min}"] & S \text{ (Stable)} \end{tikzcd}$$

Find the evolution of this decay chain by solving the coupled differential equation numerically. Assume $N_P(t=0) = N_0$, and $N_Q(t=0) = N_R(t=0) = N_S(t=0) = 0$.

(b) Consider an example where the decay chain is


$$\begin{tikzcd} A \arrow[r, "98\%"] \arrow[dr, "2\%"'] & B \arrow[r] & D \\ & C \arrow[u] & \end{tikzcd}$$

The half-life of isotopes is A : 2760 s, B : 198 s, C : 132 s, and D is stable.

- Find the system of coupled first-order differential equations that demonstrate the time evolution of the above decay chain.
- Solve the system of coupled differential equation numerically using RK4 method for the initial condition $N_A(t=0) = N_0$ and $N_B(t=0) = N_C(t=0) = N_D(t=0) = 0$.

Problem 2

Consider a cannonball shot from a cannon standing on level ground. For simplicity, assume the motion of the cannonball is restricted to the two-dimensional plane (XY) near the earth's surface, so the acceleration due to gravity is constant. Consider the air drag force, which is directed opposite to the cannonball's velocity and proportional to the velocity squared.

- Write the differential equation governing the motion of the cannonball.
- Solve these differential equations using the RK4 method for a cannonball of mass 1 Kg, with a radius of 8cm, shot at 30^0 to the horizontal with an initial speed of 100 m/s. The density of the air is 1.22 Kg/m^3, and the coefficient of drag is 0.47.

In order to make the problem more realistic, consider the following changes in the model.

- The Density of air varies with the altitude

$$\rho(y) = \rho_0 \exp\left(-y/y_0\right); \quad y_0 = 10^3 m$$

- Density of air varies with altitude and temperature.

$$\rho(y) = \rho_0 (1 - ay/T_0)^{\alpha}; \quad a = 6.5 \times 10^{-3} K/m, \ \alpha = 2.5$$

- Drag coefficient is velocity dependent.

$$C_d(v) = 0.47 \left[1 + \frac{1.5}{1 + \exp\left(\frac{v - v_d}{\Delta}\right)}\right]; \quad v_d = 35 m/s, \Delta = 5 m/s$$

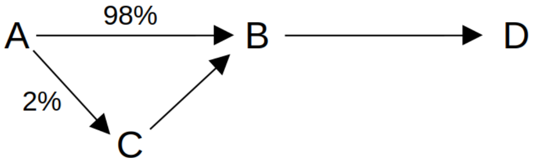
Solution 1(a)

$$P \xrightarrow{3.11\text{ min}} Q \xrightarrow{26.8\text{ min}} R \xrightarrow{19.9\text{ min}} S\ (\text{Stable})$$

The half life $t_{1/2} = (\ln 2)/\lambda \implies \lambda = (\ln 2)/t_{1/2}$
The governing equations for decay chain can be written as,

$$\frac{dN_P}{dt} = -\lambda_P N_P$$
$$\frac{dN_Q}{dt} = -\lambda_Q N_Q + \lambda_P N_P$$
$$\frac{dN_R}{dt} = -\lambda_R N_R + \lambda_Q N_Q$$
$$\frac{dN_S}{dt} = \lambda_R N_R$$

Solution 1(b)


$$\begin{tikzcd} A \arrow[rr, "98\%"] \arrow[dr, "2\%"'] & & B \arrow[r] & D \\ & C \arrow[ur] & & \end{tikzcd}$$

The half-life of isotopes is A : 2760 s, B : 198 s, C : 132 s, and D is stable. The governing equations for decay chain can be written as,

$$\frac{dN_A}{dt} = -\lambda_A N_A$$
$$\frac{dN_B}{dt} = -\lambda_B N_B + (0.98)\lambda_A N_A + \lambda_C N_C$$
$$\frac{dN_C}{dt} = -\lambda_C N_C + (0.02)\lambda_A N_A$$
$$\frac{dN_D}{dt} = \lambda_B N_B$$

Solution 2

The net force acting on cannonball is

$$\boldsymbol{F} = \boldsymbol{F}_g + \boldsymbol{F}_d = -mg\hat{\boldsymbol{y}} - \alpha|\dot{\boldsymbol{r}}|^2 \frac{\dot{\boldsymbol{r}}}{|\dot{\boldsymbol{r}}|}$$

where $\boldsymbol{r}$ and $\dot{\boldsymbol{r}}$ are respectively the position and velocity, m is the mass of the cannonball, g is the acceleration due to gravity, and $\alpha = \frac{1}{2}\rho C_d A$. C_d is the drag coefficient, A is the cross-sectional area of the projectile, and ρ is the density of the air.

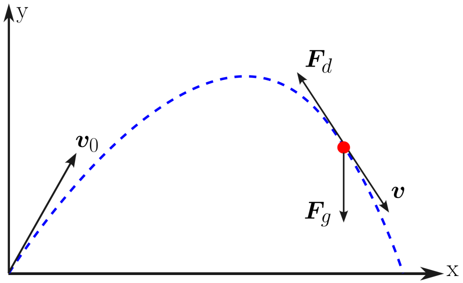


Figure: Trajectory of cannonball with air drag.

- According to the Newton's 2nd law of motion,

$$m\frac{d^2\mathbf{r}}{dt^2} = \mathbf{F} = -mg\hat{\mathbf{y}} - \frac{\rho C_d A}{2}\left[\frac{dx}{dt}\left\{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2\right\}^{1/2}\hat{\mathbf{x}} + \frac{dy}{dt}\left\{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2\right\}^{1/2}\hat{\mathbf{y}}\right]$$

- Equating the x and y components of the above equation.

$$\frac{d^2x}{dt^2} = -\frac{\pi R^2\rho C_d}{2m}\frac{dx}{dt}\left\{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2\right\}^{1/2}$$

$$\frac{d^2y}{dt^2} = -g - \frac{\pi R^2\rho C_d}{2m}\frac{dy}{dt}\left\{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2\right\}^{1/2}$$

- In the absence of air-resistance, motion of the cannonball is independent of its mass. But in real-world mass certainly makes difference.

- We can rewrite the two 2nd order ODEs into four first order coupled ODEs.

$$
\begin{aligned}
\frac{dx}{dt} &= v_x \\
\frac{dv_x}{dt} &= -\frac{\pi R^2 \rho(y) C_d(v)}{2m} v_x (v_x^2 + v_y^2)^{1/2} \\
\frac{dy}{dt} &= v_y \\
\frac{dv_y}{dt} &= -g - \frac{\pi R^2 \rho(y) C_d(v)}{2m} v_y (v_x^2 + v_y^2)^{1/2}
\end{aligned}
$$

- To determine the trajectory, use the provided initial conditions.

$$
x\Big|_{t=0} = x_0 = 0 \qquad\qquad y\Big|_{t=0} = y_0 = 0
$$

$$
v_x\Big|_{t=0} = v_0 \cos\theta \qquad\qquad v_y\Big|_{t=0} = v_0 \sin\theta
$$