1 LECTURE 01 : INTRODUCTION AND GENERAL CONCEPTS

1.1 DEFINITIONS

- **Computational Science** : a field that puts the computer at the center of the methodology, the intersection of three fields : computer science,applied mathematics and the science of the phenomena to be studied.
- **System** : something in the real world we are interested in.
- **Abstraction** : deciding which details can be left out from the system.
- **Model** : a description of the system that includes only the features we think are essential.
- **M&S** : understanding and evaluating the interaction of parts of a real or theoretical system.
- **Modelling** : the process of building a model.
- **Simulation** : The operation of a model in terms of time or space (using the model).
- **System** : something in the real world we are interested in.
- **Abstraction** : deciding which details can be left out from the system.
- **Model** : a description of the system that includes only the features we think are essential.

1.2 MODEL COMPONENTS

- **Entities** : a physical or logical entity that must be explicitly captured in your system.
- **Attributes** : characteristics of an entity.
- **State variables** : variables that are used to track a property of a static entity over time.
- **Events** : something that causes our system to change its state.
- **Activities** : actions that are performed by the system for a finite duration of time.

1.3 MODELS CLASSIFICATION

1.3.1 STATIC VS DYNAMIC MODELS

static models are used to represent systems where **time plays no role** like Monte Carlo models, **dynamic models** used to simulate models that **evolve over time** like a conveyor system in a factory.

1.3.2 DETERMINISTIC VS STOCHASTIC SIMULATION MODELS

a **deterministic model** does not contain any **probabilistic** (random) components for example : differential equations,on the other hand a **stochastic model** contains at least one **random** component for example : queuing systems.

1.3.3 CONTINUOUS VS DISCRETE SYSTEMS

Continuous system is affected by the state variable, which changes continuously as a function with time (like car moving systems),a **discrete system** is affected by the state variable changes at a discrete point of time.

1.4 MODELING SPACE AND TIME

1.4.1 TIME DIMENSION

Three ways to capture time in system :

- **Continuous Time** : only differential Equations-based models can deal with continuous Time.
- **Discrete time** : we care about the state of the system in discrete time steps : $n \times \Delta t$.
- **Discrete Event Time** : we only care about the state of the system whene certain events occur.

1.4.2 Space Dimension

Tow approaches to model the space dimension : the **Eulerian** approach and the **Lagrangian** approach :

- **Eulerian method** : you take the point of view of an observer who sits at a fixed position,space is supposed to be continuous but discretized into cells in computer models .
- **Lagrangian method** : take the point of view of the moving objects.

1.5 Common Simulation Modeling Techniques

- **Mathematical Equations (ODEs and PDEs)** : DE equations that specify the relations between the derivatives of variables.
- **Monte Carlo methods** : results are computed based on repeated random sampling and statistical analysis,used when other approaches are difficult or impossible to use.
- **Discrete Event Simulation Techniques** : Between consecutive events, no change in the system is assumed to occur.
- **Multi-Agent Modeling Technique** : Modeling the basic entities as individuals and observe the global emergent behavior.
- **Cellular Automata** : a discrete model,consists of a grid of cells, each one is of a finite number of states,the grid is updated according to some fixed rule.
- **Complex Network** : complex networks represents the interactions between a set of measurable variables,modeled using a graph.

2 LECTURE 03 : PROBABILITIES AND RANDOM NUMBER SIMULATION

2.1 PART 01 : PROBABILITIES SIMULATION

2.1.1 DEFINITIONS

- **Random experiment** : an experiment that can result in a different outcome, even when repeated under the same conditions.
- **sample space** : set of all possible outcomes,denoted by Ω.
- **Event** : a subest of Ω.
- **Probability** : quantifies the chance of each outcome.
- **Random variable** : a variable whose possible values are numerical outcomes of a random event.

2.1.2 BAYES THROEME

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

2.1.3 ASSIGNING PROBABILITIES TO OUTCOMES

when assigning probabilities to outcomes four conditions must be respected :

- $P(x_i) \in [0, 1]$
- $\sum\limits_i P(x_i) = 1$
- $P(E_j) = \sum\limits_{x \in E_j} P(x)$
- $P(\cup_j E_j) = \sum\limits_j P(E_j)$: where E_j are disjoint.

2.1.4 RELATIVE FREQUENCY AND PROBABILITY

let E be an event

$$f(E) = \frac{v}{n}$$

where n is the number of experiments and v is the number of occurrences of the event.

For a large number of experiments :

$$f(E) \approx P(E)$$

2.1.5 PROBABILITY DISTRIBUTIONS

DISCRETE PROBABILITY DISTRIBUTIONS :

The probability mass function of X specifies function of X is a function $P(x) \equiv P(X = x)$.

The probability mass function satisfies the following conditions :

- $\forall x \in \Omega : 0 <= P(x) <= 1$.
- $\sum\limits_{x \in \Omega} P(x) = 1$

Mean and variance :

$$\mu = \sum_x x \times P(x)$$

$$\sigma^2 = \sum_x (x-\mu)^2 \times P(x)$$

Definitions

Distribution	Mass function	mean	variance
The Binomial Distribution	p(x) = $\begin{cases} \binom{t}{x} \times p^t \times (1-p)^{t-x} \text{ if } x \in \{0,1,..,t\} \\ 0 \text{ else} \end{cases}$	$t \times p$	$t \times p \times (1-p)$
The Geometric Distribution	p(x) = $\begin{cases} p(1-p)^{x-1} \text{ if } x \in \{1,..\} \\ 0 \text{ else} \end{cases}$	$\frac{1-p}{p}$	$\frac{1-p}{p^2}$
The Poisson Distribution	p(x) = $\begin{cases} \frac{e^{-\lambda}\lambda^x}{\lambda!} \text{ if } x \in \{0,1,..\} \\ 0 \text{ else} \end{cases}$	λ	λ

CONTINUOUS PROBABILITY DISTRIBUTIONS :

Described by a **probability density function** $f(x)$.

The probability density function satisfies the following conditions :

- $f(x) >= 0$
- $\int_{-\infty}^{\infty} f(x) = 1$
- $P(a < x < b) = \int_a^b f(x)$

Mean and variance :

$$\mu = \int_{-\infty}^{\infty} x \times f(x)$$

$$\sigma^2 = \int_{-\infty}^{\infty} (x-\mu)^2 \times f(x)$$

Definitions

Distribution	Density function	mean	variance
The Uniform Distribution	p(x) = $\begin{cases} \frac{1}{b-a} \text{ if } x \in [a,b] \\ 0 \text{ else} \end{cases}$	$\frac{a+b}{2}$	$\frac{(b-a)^2}{12}$
The Exponential Distribution	p(x) = $\begin{cases} \frac{1}{\beta}e^{-\frac{x}{\beta}} \text{ if } x >= 0 \\ 0 \text{ else} \end{cases}$	β	β^2
The Normal Distribution	$\frac{1}{\sqrt{2\pi\sigma^2}}e^{\frac{-(x-\mu)^2}{2\sigma^2}}$	μ	σ^2

2.2 Part 02 : Random number simulation

Random random are generated using deterministic algorithms called pseudo random number generators (PRNGs) controlled by seed,which means that same seed will always yield the same results.

2.2.1 Random number simulation with python

```
import random
print(random.random())
```

Figure 1: Generate a random float between 0 and 1.

```
import random

random.seed(1)
print(random.random())
```

Figure 2: Reproduciblity : the above code multiple times always yields the same result

```
import random
print(random.unifrom(1,100))
```

Figure 3: Generate a random number in the range $[a, b)$

```
import random
print(random.randint(-100,100))
```

Figure 4: Generate a random integer in the range $[a, b]$

```
import random
print(random.choice([1,2,3]))
```

Figure 5: sampling an element from a list

```
import random
print(random.sample([1,2,3], k=2))
```

Figure 6: sampling multiple elements from a list,this doesn't select the same element twice.

```
import numpy as np
print(np.random.uniform(a,b,N))
```

Figure 7: sample N elements from the interval $[a, b)$

```
import numpy as np
print(np.random.binomial(t,p,N))
```

Figure 8: sample N elements from the binomial distribution of parameters t and p.

```
import numpy as np
print(np.random.normal(mu,sigma,N))
```

Figure 9: sample N elements from the binomial normal distribution of parameters mu and $sigma$.

3 Lecture 04 : Modeling Dynamical Systems

3.1 Simulating Discrete-Time Systems

3.1.1 First-order linear systems :

First order : x_t depends only on x_{t-1}

$$x_t = f(x_{t-1}, t)$$

Linear : f expression is linear combination of $x_{t-1}, x_{t-2}, ...$

First order linear system :

$$x_t = a.x_{t-1} + b$$

The general solution is given by :

$$x_t = a^t.x_0 + \begin{cases} \frac{a^n - 1}{a-1} \text{ if } a \neq 1 \\ n.b \text{ otherwise} \end{cases}$$

3.1.2 Solving first-order linear systems with python:

```
from sympy import Function,rsolve
from sympy.abc import t

n0 = 1
a = 1.1
b = 2

N = Function('N')
f = N(t+1) - (a * N(t) + b) # N(t+1) = a * N(t) + b

sol = rsolve(f, N(t), {N(0) : n0})
n5 = sol.subs(t, 5)

print(sol)
print(f"{n5:.2f}")
```

Figure 10: Solving First order linear systems with python

```
Nt = 59
t = np.linspace(0, (Nt+1), (Nt+2))
E = n0 * np.power(a, t)

N = np.zeros(Nt+2)

N[0] = n0

for i in range(1, Nt+2):
   N[i] = a * N[i-1]

fig, (ax1, ax2) = plt.subplots(ncols=1,nrows=2)

ax1.plot(t, E, label='Exact solution')
ax1.scatter(t, N, label='Numerical solution', color='red', s=5)

ax2.plot(t, (N - E) ** 2, label='Err(t)')

ax1.legend()
ax2.legend()

plt.show()
```

Figure 11: Numerical vs Exact solution

3.1.3 SIMULATING CONTINUOUS TIME DYNAMICAL SYSTEMS

ORDINARY DIFFERENTIAL EQUATIONS :

$$f_i(x_i, \dot{x}_i, \ddot{x}_i, ...; t) = 0$$

Characteristics :

- **Order** : the highest derivative order.
- **Dimension** : the dimension of the vector x.
- **Autonomous** : f_i doesn't depend on t.
- **Linear** : Linear combination of $\frac{d^k x}{dt^k}$.

1D AUTONOMOUS EQUATION :

$$\frac{dx}{dt} = f(x)$$

Exact Solution :

$$\int \frac{dx}{f(x)} = t + c$$

if $\int \frac{dx}{f(x)}$ is impossible/hard to find $\implies$ solve numerically.

Examples :

Name	Equation	Solution
exponential population growth	$\frac{dx}{dt} = x$	$C_1 e^t$
logistic population growth	$\frac{dx}{dt} = x(1-x)$	$\frac{1}{C_1 e^{-t}+1}$

Equilibrium positions :

- unstable equilibrium : the derivative is equal to zero (constant curve) but the curve converges.
- stable equilibrium : the derivative is equal to zero (constant curve) but the curve does not converges.

SOLVING 1D AUTONOMOUS EQUATIONS WITH PYTHON:

Example : exponential population growth

```
import sympy

t = sympy.symbols('t')
x = sympy.Function('x')

equation = x(t).diff(t) - x(t)

sol = sympy.dsolve(equation, x(t))

print(sol)
```

Figure 12: Solving 1D autonomous equations with python : Exact solution

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

x0 = 1.0
t = np.linspace(0,100,100)

def diff(x, t):
   return x

y = odeint(diff, x0, t)

plt.plot(t, y)
```

Figure 13: Solving 1D autonomous equations with python : Numerical solution

2D AUTONOMOUS EQUATIONS :

$$\begin{cases} \frac{dx}{dt} = f(x, y; t) \\ \frac{dy}{dt} = g(x, y; t) \end{cases}$$

Lotka-Volterra equations (Predator-prey equations) :

$$\begin{cases} \frac{dx}{dt} = ax - bxy \\ \frac{dy}{dt} = -cy + dxy \end{cases}$$

where a, b, c and b are assumed positive.

Van der Pol oscillator :

$$\ddot{x} - a(1 - x^2)\dot{x} + x = 0$$

$\mid x \mid > 1$: loses energy else absorbs energy.

after applying first order reduction the equation can be rewritten as follows :

$$\begin{cases} \dot{x} = y \\ \dot{y} = a(1 - x^2)y - x \end{cases}$$

4 LECTURE 05 : SOLVING PROBLEMS WITH MONTE-CARLO SIMULATION TECHNIQUE

4.1 LLN THEOREM

Let X_n be an independent, and identically distributed (i.i.d.) sequence sampled from any probability distribution P with mean μ Then:

$$\hat{\mu}_n = \lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n} X_k = \mu$$

Error Estimation :

$$\mid \hat{\mu}_n - \mu \mid = \frac{\sigma}{\sqrt{n}}$$

4.2 THE CENTRAL LIMIT THEOREM

Let X_n be an independent, and identically distributed (i.i.d.) sequence sampled from any probability distribution P with mean μ and variance σ^2 . Then:

$$\hat{\mu}_n \xrightarrow{n\to\infty} N(\mu, \sigma^2)$$

Confidence Interval :

$$\hat{\mu}_n \pm 1.96 \frac{\sigma}{\sqrt{n}} = [\hat{\mu}_n - 1.96 \frac{\sigma}{\sqrt{n}}, \hat{\mu}_n + 1.96 \frac{\sigma}{\sqrt{n}}]$$

4.3 MONTE CARLO SIMULATION: BASIC STEPS

- Define possible inputs.
- Generate inputs randomly.
- Deterministic computations on the input.
- aggregate the results.

4.4 MONTE-CARLO APPLICATIONS IN RESOLVING NUMERICAL INTEGRALS

The task is to evaluate :

$$A = \int_I f(x)$$

Steps :

- Sample points x randomly from I.
- $A = interval\_length \times mean$ of $f(x)$

4.5 Monte-Carlo Applications to Uncertainty Analysis

The task is to : quantify uncertainties propagated in models variables.

Example :

Let :

$$X \text{ follows } Uniform([0,1])$$
$$Y \text{ follows } Exponential(\beta = 2.25)$$

And :

$$Z = X * Y$$

Q : what is the 95^{th} percentile of Z ?

Answer :

Sample N points from from X and Y's distribution and compute their product to produce new N samples and plot the histpgram of Z.