AERIAL ROBOTICS: WEEK 03

Welcome to Week 3 of the Aerial Robotics course!

This week, we'll start with a fun topic, **an introduction to the agility and maneuverability** of quadcopters, primarily concentrating on their kinematics and dynamics.

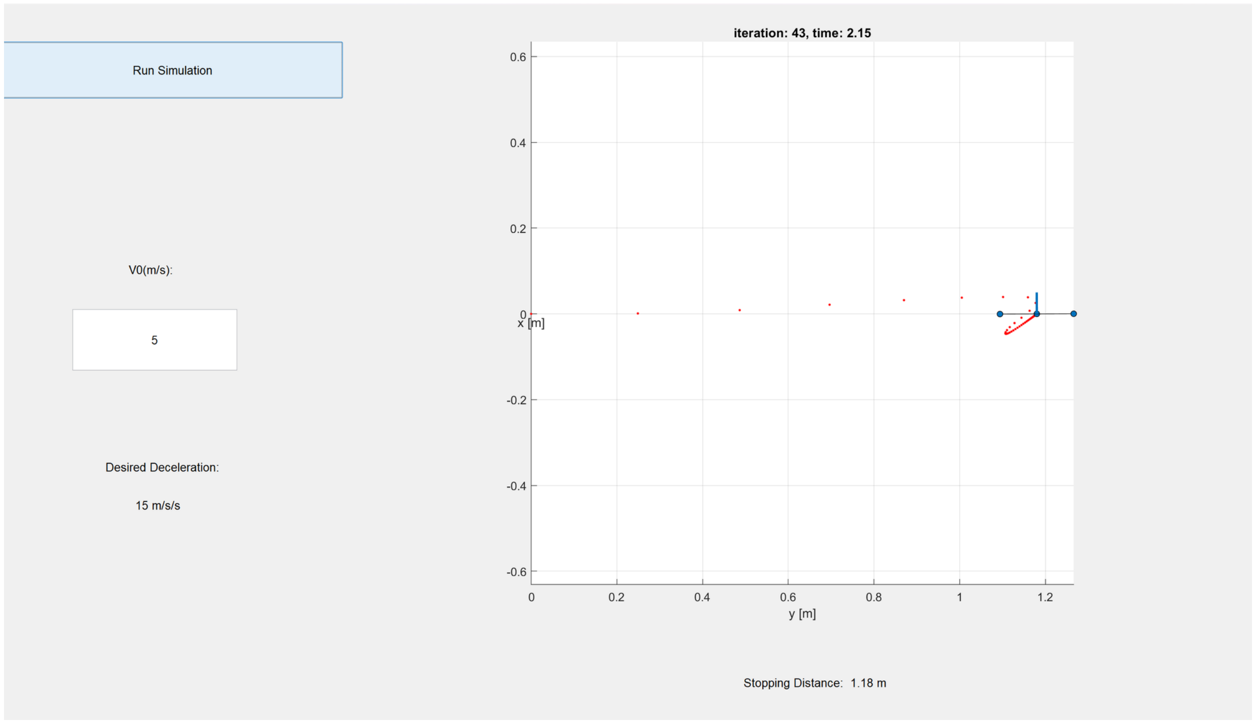
- **Agility and maneuverability**

https://youtu.be/5dZ411mq_Ow

Download the simulation GUI attached below and unzip its content to the directory you wish to complete the exercise. Open MATLAB in that directory and type "*runsim*" in MATLAB Command Window to start the GUI.

StoppingDistanceExercise.zip

You can see the stopping distance as well the time taken for various values of v0 (initial velocity) for a deceleration of 15 m/s^2. To gain a sense of the simulation, experiment with it a little.



- **Dynamical Systems**

Dynamical systems are mathematical objects used to model physical phenomena whose state (or instantaneous description) changes over time. These models are employed in a variety of applications, including environmental modeling, industrial equipment diagnostics, medical diagnosis, and financial and economic forecasting.

Imagine you're piloting a quadrotor drone through a 3D obstacle course. To precisely navigate and control its flight, how many state parameters would you need to fully describe the motion of the quadrotor in three-dimensional space? Perhaps the answer will reveal itself by the end of this week's content!

https://youtu.be/CMKp6OWk78w

Picture this: There's an example below showcasing a 2 DOF (degree of freedom) system. While it's not an exercise, it offers a glimpse into the fascinating world of controls. Though you may need some foundational knowledge to fully grasp what's happening (which you'll gain this week), it's a neat way to get a 'feel' for controls. Dive in and take a look—it's an exciting preview of what lies ahead!

Control Tutorials for MATLAB and Simulink - Inverted Pendulum: PID Controller Design

Key MATLAB commands used in this tutorial are: tf , impulse , feedback , pid

https://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum&section=ControlPID

```
M = 0.5;
m = 0.2;
b = 0.1;
I = 0.006;
g = 9.8;
l = 0.3;
q = (M+m)*(I+m*l^2)-(m*l)^2;
s = tf('s');
P_pend = (m*l*s/q)/(s^3 + (b*(I + m*l^2))*s^2/q - ((M + m)*m*g*
Kp = 100;
Ki = 1;
Kd = 20;%you're encouraged to twirk these values and see how th
C = pid(Kp,Ki,Kd); %Note you won't know these functions as of y
T = feedback(P_pend,C);
t=0:0.01:10;
impulse(T,t)
title({'Response of Pendulum Position to an Impulse Disturbance
```

Hopefully, by tweaking the example, you've noticed that it's quite similar to the 1 DOF controller we previously created. This realization might just be the key to understanding the nuances of control systems

This video will guide you through all you need to know about inverted pendulum.

https://youtu.be/qjhAAQexzLg

- **Quadrotor Dynamics**

To understand the dynamics of a quadrotor, such as its equations of motion and moments, we first need to grasp the kinematics of the quadrotor. This requires some basic mathematical knowledge.

When tracking a drone's motion, we need to handle data in two different frames of reference: the Earth (or Inertial) Frame and the Body Frame.

https://youtu.be/Xeemdai9e9Y

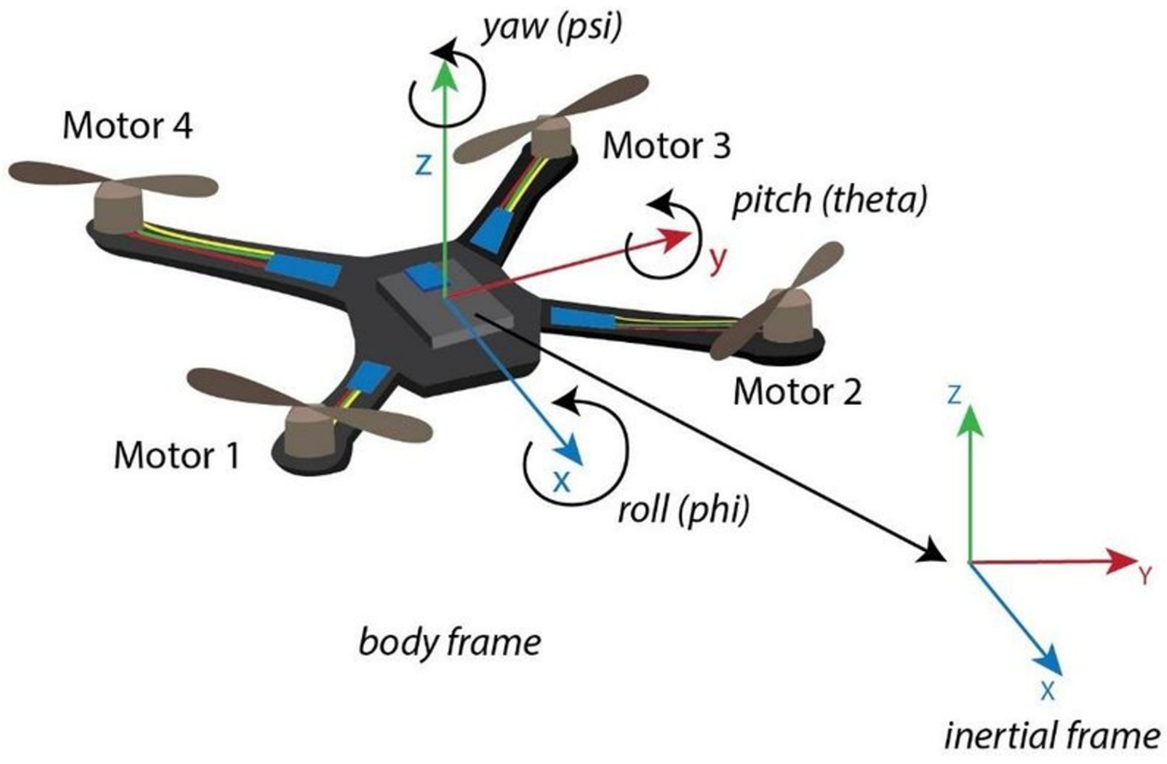
The angles θ (theta), φ (phi), and ψ (psi) that describe the drone's orientation are usually measured using the drone's onboard Inertial Measurement Unit (IMU). This sensor measures how fast the drone's body is rotating around its own axes and provides this angular velocity as its output. It's important to note that these angular velocities are with respect to the drone's Body Frame, not the Earth Frame.

- * In the context of a quadrotor or drone, the angles θ (theta), φ (phi), and ψ (psi) represent the orientation of the drone in 3D space. These angles are known as Euler angles, and they describe the drone's rotation around its three principal axes where:

1. **Theta (θ) - Pitch Angle**
2. **Phi (φ) - Roll Angle**
3. **Psi (ψ) - Yaw Angle**

They help in understanding and implementing the drone's orientation and movement in 3D space. *

When processing the outputs from the IMU, we need to be careful and understand that the angular velocities are relative to the drone's own axes (Body Frame) rather than the fixed Earth Frame.



The diagram above shows both the Earth (Inertial) Frame and the Body Frame for reference.

To convert angular data from the Body Frame to the Inertial Frame, we use a Rotational Matrix.

Before we get into rotational matrices, I recommend watching the following video by 3Blue1Brown to get a better intuition about what we are doing. While our case is very specific, the video will help clarify many concepts. After watching the video, try the small example below to better understand 3D transformations. This will be helpful later on.

https://youtu.be/kYB8IZa5AuE

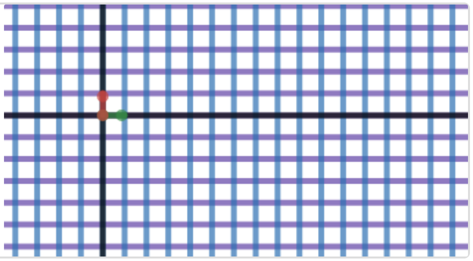
```
%Type in the transforming matrix below: (One is already given)
clf
set(gcf,'Visible','on');
grid on;
M = [1/sqrt(2) -1/sqrt(2);1/sqrt(2) 1/sqrt(2)] % a note here jus
title({''});
xlabel('');
ylabel('');
T = [1 0;0 1]; %initialization
for t = 0:200
    T(1,1) = ((200-t)*(1-M(1,1))/200)+M(1,1);
    T(2,1) = ((200-t)*(-M(2,1))/200)+M(2,1);
    T(1,2) = ((200-t)*(-M(1,2))/200)+M(1,2);
    T(2,2) = ((200-t)*(1-M(2,2))/200)+M(2,2);
    init1 = T*[-5;100];
    init2 = T*[-5;-100];
    hold on
    plot([init1(1),init2(1)],[init1(2),init2(2)]);
    quiver(0,0,T(1,2),T(2,2),1,'r','linewidth',2);
    quiver(0,0,T(1,1),T(2,1),1,'r','linewidth',2);
    for i = -4:5
         xlim([-5,5]);
    ylim([-5,5]);
        plot([(T(1,1)*i+100*T(1,2)),(T(1,1)*i-100*T(1,2))],[(T(
    end
    for i = -4:5
         xlim([-5,5]);
    ylim([-5,5]);
        plot([(T(1,1)*100+i*T(1,2)),(-T(1,1)*100+i*T(1,2))],[(T
    end
    xlim([-5,5]);
    ylim([-5,5]);
    pause(.1);
    clf
end
hold on
for i = -4:5
    plot([(M(1,1)*i+100*M(1,2)),(M(1,1)*i-100*M(1,2))],[(M(2,1)
end
for i = -4:5
    plot([(M(1,1)*100+i*M(1,2)),(-M(1,1)*100+i*M(1,2))],[(M(2,1
end
hold off
xlim([-5,5]);
ylim([-5,5]);
```

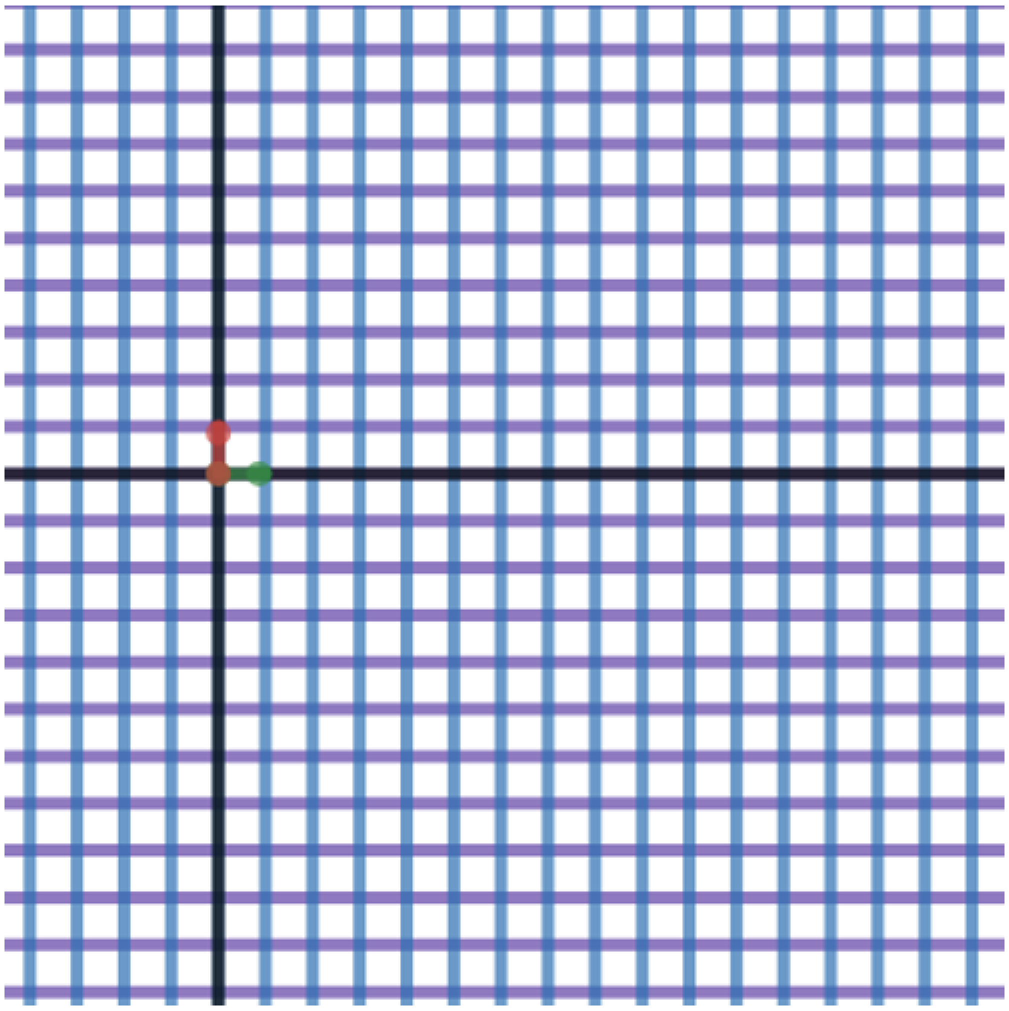
Hope this gives you some insight! If you want to experiment more, visit this link.

2D linear transformation

Explore math with our beautiful, free online graphing calculator. Graph functions, plot points, visualize algebraic equations, add sliders, animate graphs, and more.

https://www.desmos.com/calculator/upooihuy4s



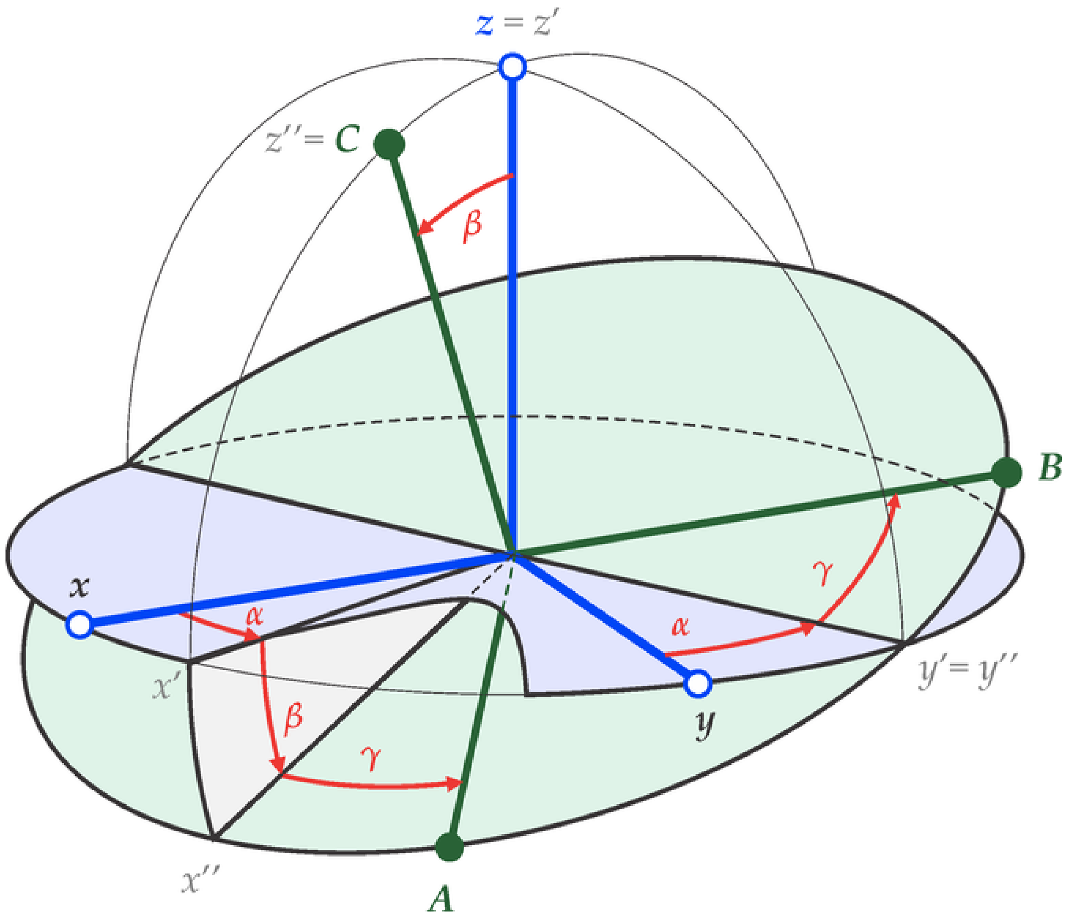


Finally, we hope this clears up 2D rotation for you. Now, let's shift to 3D rotations…

https://youtu.be/WhEf0lvGCIM

EULER’S ANGLES

Euler showed that three coordinates are necessary to describe a general rotation, and these coordinates are called the Euler angles.



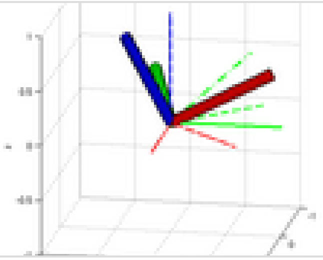
https://youtu.be/3Zjf95Jw2UE

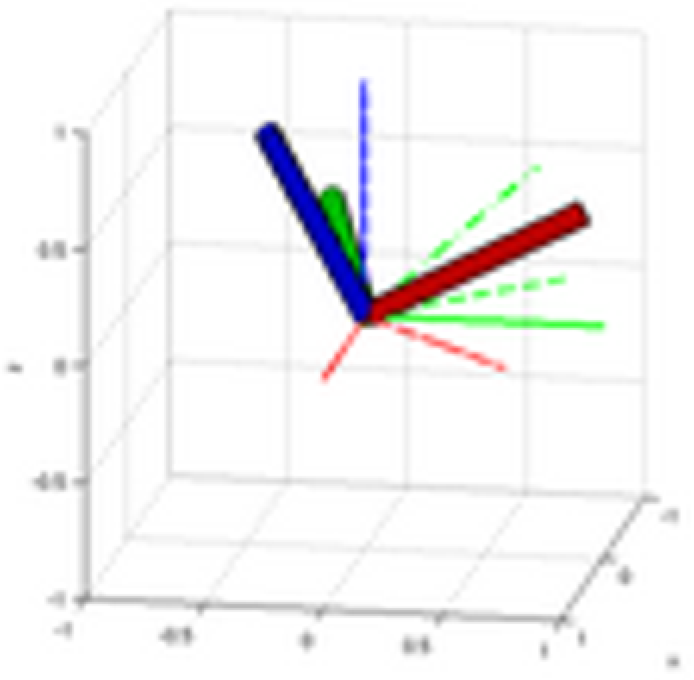
To understand Euler angles notation a bit better, let's try to replicate what we did with 2D rotations. We'll give you an example of ZXY rotation. We were about to code an animation for this, but it looks like someone has already done it, so we are attaching it here.

animEuler

Animate sets of Euler rotations using arbitrary Euler angles and rotation order.

https://in.mathworks.com/matlabcentral/fileexchange/23964-animeule
r





Download animEuler, open it and read the comments carefully to operate it, feel free to play with it!

We've seen two different approaches to rotations. The number of unknowns in these rotations is called 'coordinates'. For example, using rotation matrices, you can represent rotations with a set of nine numbers or coordinates, but six of those are redundant. This means we can actually use any three coordinates to represent the rotation (these are the Euler angles). With Euler angles, you get a set of three numbers or coordinates.

Now, let's look at another way to represent rotations, which involves explicitly specifying the axis of rotation and the angle of rotation.

https://youtu.be/zrDCI89bSp4

Having become familiar with the concept of rotations and displacements, it's time to think about the rate of change of rotations, leading us to the concept of an angular velocity vector. But what does it mean to differentiate a rotation and get a velocity?

https://youtu.be/HBLZuV92qGw

Here's a numerical example involving rigid body displacements.

https://youtu.be/4XcoUhpdUjw

In the example video, you'll notice a lot of equations. MATLAB can be very helpful if you want to quickly verify a few results.

```
syms theta
R = [cos(theta) -sin(theta) 0;sin(theta) cos(theta) 0; 0 0 1]
diff(R)
R.'
simplify(R.'*diff(R))
```

You know that functions have two important properties, i.e. one to one and onto.

https://youtu.be/ks6B8RuHarw

Now that we have looked at positions and velocities (Kinematics) , it's time to study the dynamics of a quadrotor.

Formulation

https://youtu.be/kVRYanVeWq0

Newton Euler Equation

https://youtu.be/poiTKrgaHqQ

Principal axes and moment of inertia

https://youtu.be/WPM46QeOD1g

Now to track drone's movements, we keep track of its state vector *X* and its derivative.

As our drone has 6 degrees of freedom, we usually track it by monitoring these six parameters along with their derivatives (how they are changing with time) to get an accurate estimate of the drone's position and velocity of movement.

We do this by maintaining what is often called a state vector.

$\boldsymbol{X} = [x, y, z, \varphi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, p, q, r]$

Here's what each component represents:

- **x, y** and **z** are positions of the drone in the Inertial frame.
- $\dot{x}, \dot{y}, \dot{z}$ are positional / linear velocities in the inertial frame.
- φ,θ,ψ represent drone attitude / orientation in the Inertial frame.
- **p, q, r** are angular velocities in the body frame whereas.
- The derivatives of p,q,r represent the angular acceleration in the body frame.

Equations of motion

https://youtu.be/R1HPPyolmNw

Till Now, we saw the dynamical equations with a quadrotor written as matrix equations, which we call state-space form. Now, we'll see how to transform the ordinary differential equations representing a dynamical system into state-space form.

https://youtu.be/Ia0lgq44azI

If you have difficulty with digesting the state space formulation have a look at the example, we had given you (and the assignment) in week 1 for solving higher order ode, that very clearly explains how powerful state space formulations are.

This far you have developed planar and three-dimensional dynamic models of the quadrotor.

So, this week we have understood some important concepts, Dynamics of Quadrotor and math of it. Congratulations for reaching till this point of the Aerial Robotics course. Soon, we will be diving into control in the 3-dimensional space and an Introduction to Path Planning.

Until then … Happy Learning! :)

AEROMODELLING CLUB
IIT BOMBAY