**Documentation**

**6 Degrees-of-Freedom Trajectory Simulator**

**Cambridge University Spaceflight**

Contents

[1 Coordinate systems 2](#_Toc56756606)

[1.1 Inertial 2](#_Toc56756607)

[1.2 Launch Site 3](#_Toc56756608)

[1.3 Body 4](#_Toc56756609)

[2 Step by step of the simulation process 5](#_Toc56756610)

[3 Theory 6](#_Toc56756611)

[3.1 Accelerations 6](#_Toc56756612)

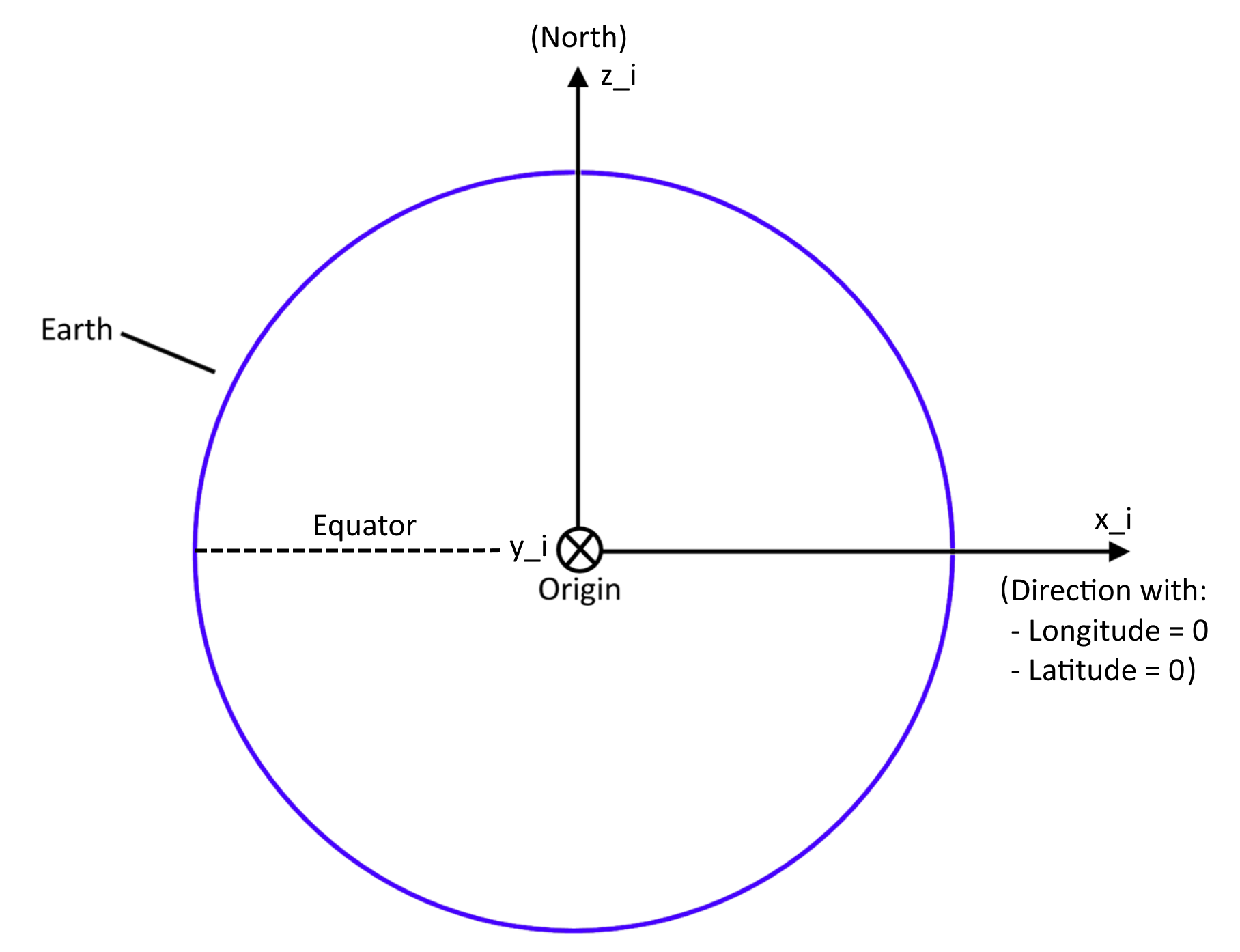
# Coordinate systems

There are three coordinate systems used in the code, all shown below.

## Inertial

The inertial coordinate system used is an Earth-centred inertial (ECI) frame.

Variables that are in inertial coordinates are denoted by the subscript ‘i’ in the code. E.g. position\_i, velocity\_i.



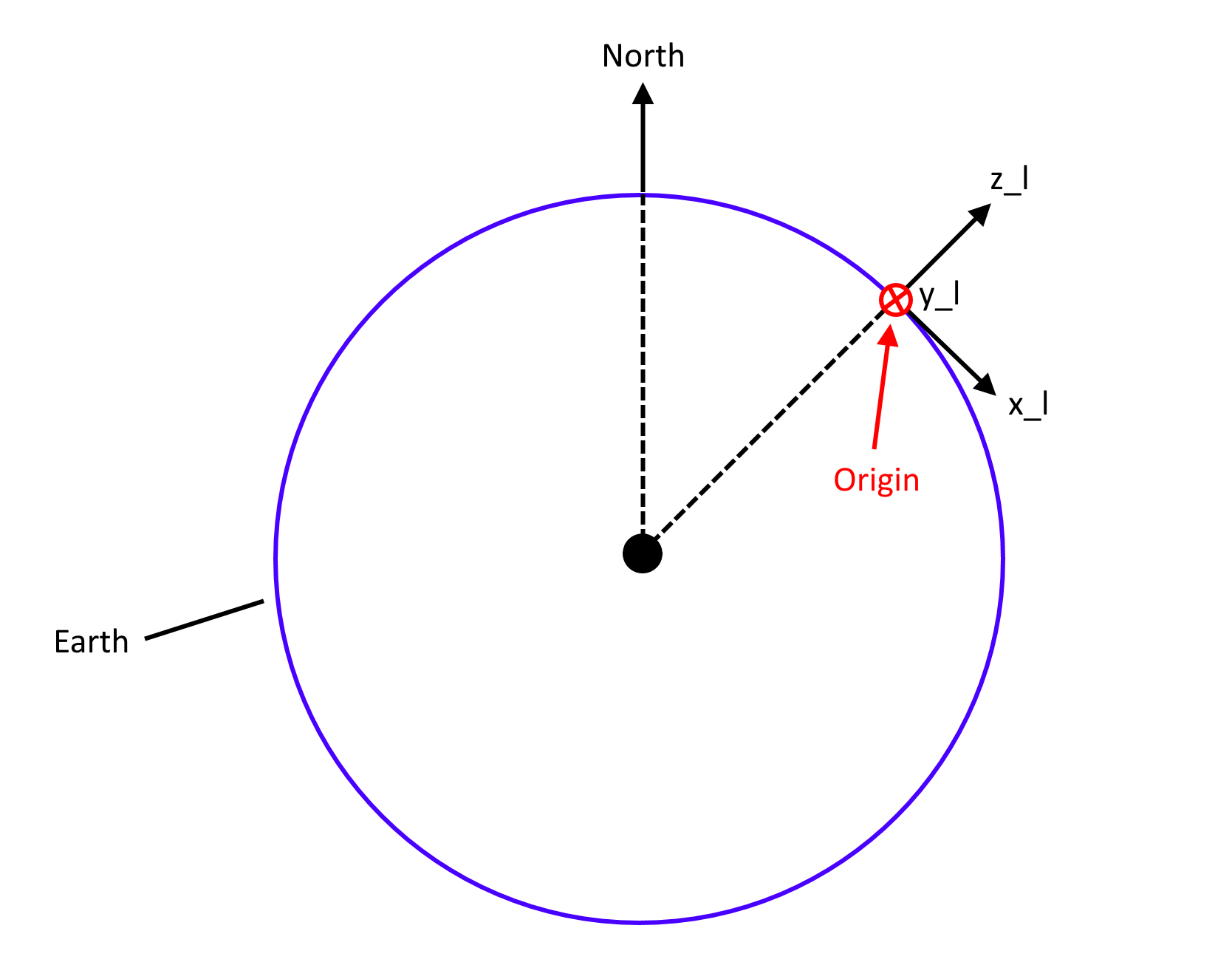
The origin is at the centre of the Earth.

* X points in the direction of 0 longitude, 0 latitude, at the start of the simulation (t=0).
* Z points North.
* Y is set by X and Z, given that it’s a right-hand coordinate system.

## Launch Site

This is a coordinate system that follows the launch site. Note that it is not inertial – it rotates with the Earth. Velocities in this reference frame are measured relative to the launch site – i.e. relative to the surface.

Variables that are in inertial coordinates are denoted by the subscript ‘l’ in the code. E.g. position\_l, velocity\_l.



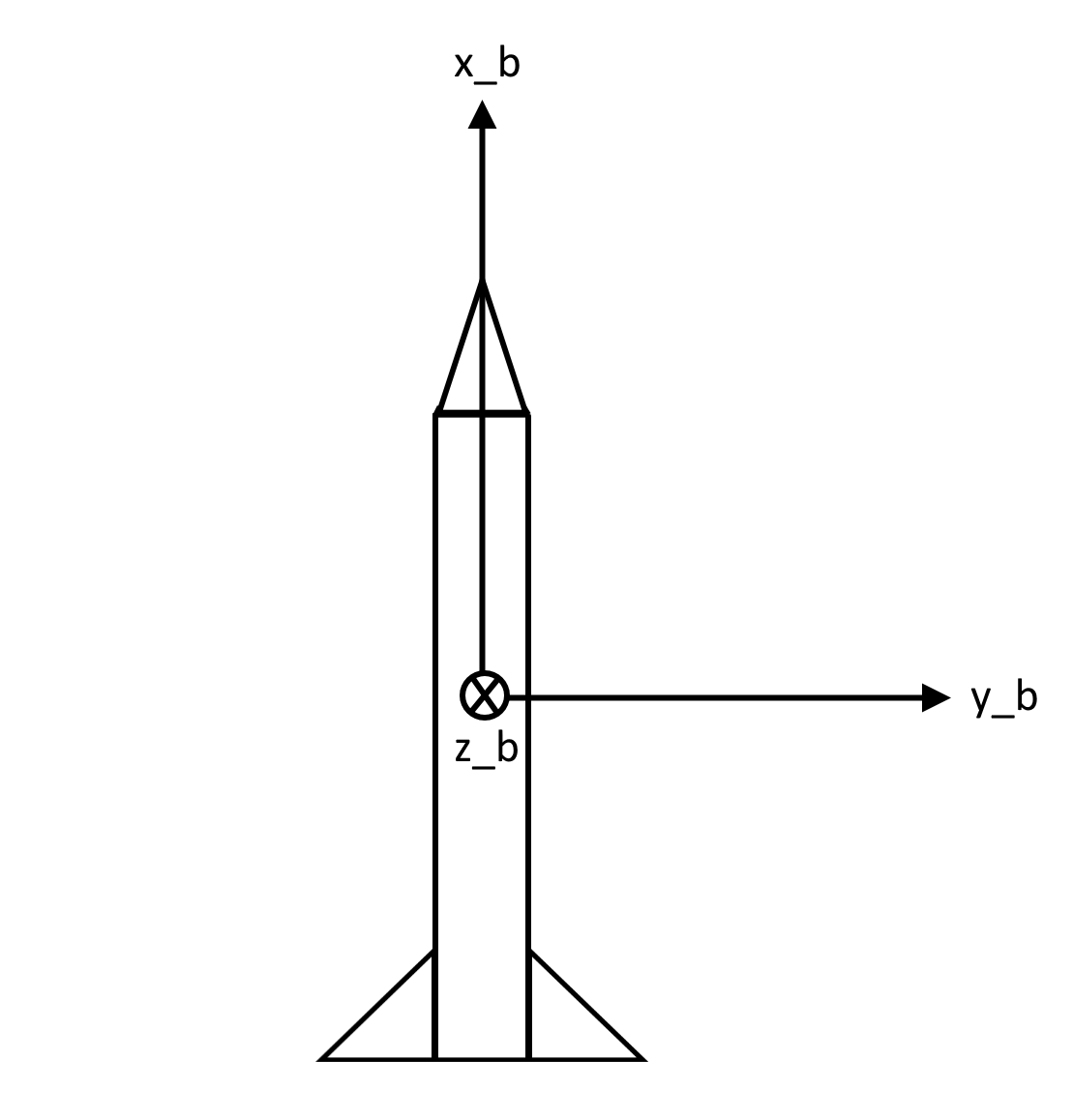
The origin is fixed to the launch site, and moves with it as time progresses.

* Z is normal to the surface of the Earth, pointing away from the centre of the Earth.
* X is pointing southwards, tangential to the surface of the Earth.
* Y is set by X and Z since it’s a right-hand coordinate system.

## Body

The body coordinate system is fixed to the rocket and rotates with the rocket.

Variables that are in inertial coordinates are denoted by the subscript ‘b’ in the code. E.g. position\_b, velocity\_b.



The chosen spot for the origin is irrelevant and is not used in any calculations. Only the directions of the body coordinate system are ever used.

* X points “forwards”, i.e. towards the nose.
* Z and Y are chosen so that they align with the launch site coordinate system, at the start of the simulation (i.e. when t=0).

# Step by step of the simulation process

Create the Rocket object. This requires creating the following ‘sub’-objects:

* **MassModel** – to model the moments of inertia, mass and dimensions.
* **Motor** – to model the propulsion system for the rocket.
* **LaunchSite** – to specify the launch site’s parameters (longitude, latitude, launch rail parameters).
* **aerodynamic\_coefficients** – currently the only option for this is to create a **RasAeroData** object. This specifies the information used for aerodynamic force and moment calculations.

output = run\_simulation(Rocket object)

Programme internally runs Rocket.step() on a loop until something triggers the end of the simulation.

Calculate accelerations using self.acceleration().

Use RK4 to integrate.

For each execution of Rocket.step()

self.accelerations():

1. Calculate forces:
   1. self.thrust() – returns thrust as a vector\_b
   2. self.aero\_forces() – returns aero forces as a vector\_b, and the position where they act (the centre of pressure).
   3. self.gravity() – returns the gravity force, as a vector\_i
2. Calculate moments about the centre of gravity
3. Calculate linear accelerations and angular accelerations, then return them

# Theory

## Accelerations