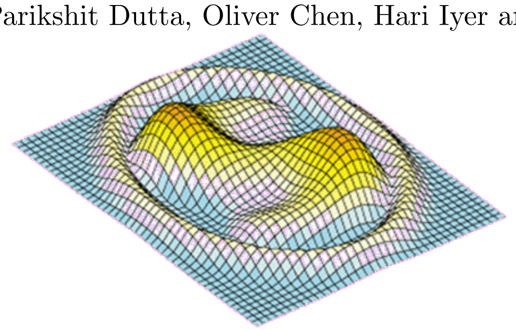
The National Airspace Trajectory-Prediction System

Algorithm Glossary

Report prepared under NASA University Leadership Initiative Project

Dr. P.K.Menon, Dr. Parikshit Dutta, Oliver Chen, Hari Iyer and Dr. Bong-Jun Yang



Optimal Synthesis Inc., Suite 240,
Los Altos, CA, 94022.

August 20, 2019

Contents

Chapter 1

Introduction

Overview of NATS and the interfaces and what will the book contain.

Chapter 2

Overall Architecture

The NATS software is designed to simulate ensemble of aircraft trajectories starting from the departure gate to the arrival gate. It is designed on a server-client framework. The server hosts all the databases, models and algorithms. The client connect with the server and invokes the simulation. Once the simulation is over, the client can access simulation outputs for post processing.

At its core, the NATS consists of three interfaces: equipment interface, environment interface and entity interface. Each interface interacts with other to create the gate-to-gate air traffic simulation. Further, at any given time the overall system-wide safety of the National Airspace System (NAS) can be investigated via a safety metric interface.

This Chapter describes the overall NATS architecture and its critical components.

2.1 The Server-Client Model

The NATS trajectory propagation engine is based on a Server-Client model. The server contains the aircraft performance models, the airspace procedures, terrain, wind and weather data as well as the algorithms for propagation. The client creates flight plans for aircraft to be simulated and invokes the simulation by sending the flight plan data to the server. The client is available to users and can be used on multiple platforms such as Python, Java™and Matlab®.

The basic structure of NATS is shown in Figure 2.1. A remote server, which hosts NATS models and databases, can be accessed by users through NATS client. The client inputs flight plans and initial states of the flights to be simulated. It is sent to the remote server where the NATS simulation is run. The trajectory outputs can be then visualized in the client side using interface of the user's choice.

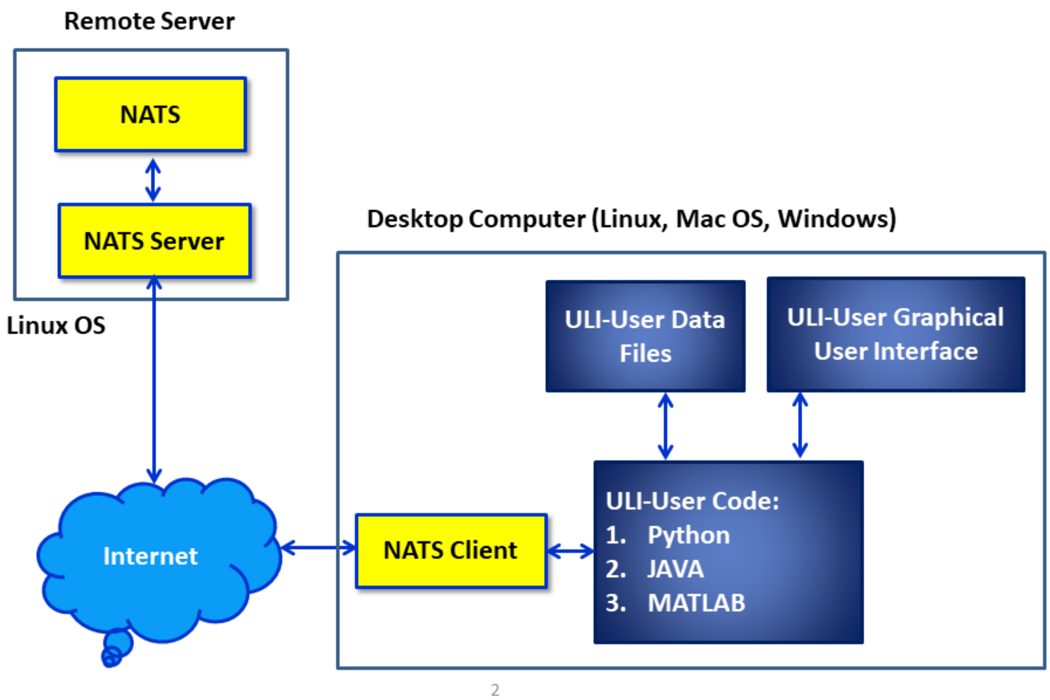


Figure 2.1: The Basic Structure of NATS.

The projected final structure of NATS is shown in Figure 2.2. Multiple users will be able to connect with NATS from different places at the same time. Further, real-time simulations with feedback from ADS-B can be inputted to NATS. Moreover, users will be able to connect virtual aircraft simulators such as x-planes to NATS. The server then would take all the user inputted aircraft and simulate them as an ensemble.

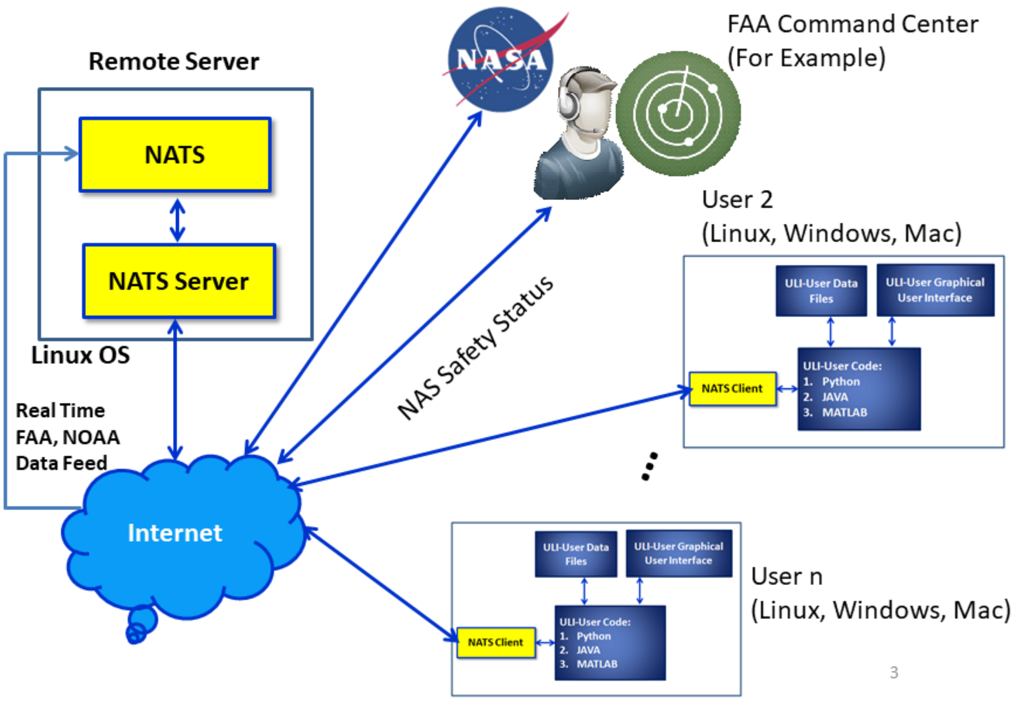


Figure 2.2: The Projected Capabilities of NATS.

2.2 NATS Interfaces

NATS software module is consists of three interfaces which interact with each other:

- **Equipment Interface**: Consists of the aircraft itself, communication and navigation systems, ground vehicles, flight deck systems etc.

- **Environment Interface**: Consists of the airports, terrain, surface winds, weather and the operating procedures laid out by FAA.

- **Entity Interface**: Consists of the pilots, controllers and the ground oeprating staff.

The entire system has been described in Figure 2.3.The NATS sofware framework is designed such that these contributing modules interact with each other to create a gate-to-gate air traffic simulation software.

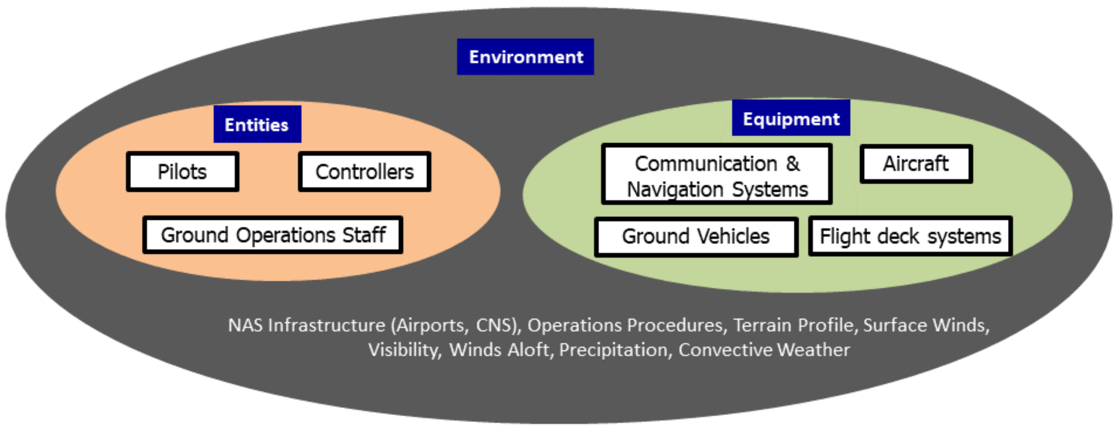


Figure 2.3: Different Components of the NATS Software.

The NATS software allows the user to access individual members of each interface and modify their properties before and during simulation.

2.2.1 Equipment Interface

As indicated in the Introduction section, the main equipments employed in the NAS are aircraft together with flight-deck automation system, ground service vehicles, communication systems, and surveillance systems. Although it may be desirable to incorporate high-fidelity models describing their nominal and off-nominal behaviors, this may not be feasible due to large number of subsystems that needs to be included, leading to unmanageable computational effort. Consequently, simpler models capturing the essential features of their operation are included. The following subsections provide additional details.

Aircraft and Ground Vehicle Models

High-fidelity aircraft models are generally described a set of 12 first-order nonlinear differential equations. However, six of these are not relevant to the system safety analysis, because they deal with the attitude motions of the aircraft in flight. Eliminating these equations from the aircraft dynamics is equivalent to assuming that the aircraft are always in moment equilibrium. Under this assumption the aircraft dynamics can be described by six first-order nonlinear ordinary differential equations, commonly termed as the point-mass model. These equations generally require detailed knowledge about the aircraft engines, aerodynamics and mass. Since it is nearly impossible to get accurate data for every aircraft that are operating in the NAS, a further simplification is made in the analysis. The simplification consists of employing the three differential equations describing the position kinematics of aircraft, together with rate of climb, rate of descent, and airspeed or Mach number bounds from well-known databases such as BADA [1]. This database was assembled by Eurocontrol using actually observed trajectories in the European airspace.

With these simplifications, the equations of motion for an aircraft are given by:

$$\dot{h} = f(h, A) \tag{2.1}$$

$$\dot{\lambda} = \frac{1}{R_e + h}\left(V\cos\gamma\cos\chi + W_N\right) \tag{2.2}$$

$$\dot{\tau} = \frac{1}{(R_e + h)\cos\lambda}\left(V\cos\gamma\sin\chi + W_E\right) \tag{2.3}$$

where, h is the altitude of the aircraft, λ is the latitude and τ is the longitude. Further, the instantaneous flight path angle can be found from $\gamma = \sin^{-1}(\dot{h}/V)$, where $V_{\min} \leq V(h, A) \leq V_{\max}$.

The control variables in this model are the airspeed V, flight path angle γ or the altitude rate $f(h, A)$, and the course angle χ. Note that the limits on the climb and descent rates and the airspeed are specified in BADA for over 400 Aircraft types. In the present NATS framework, these variables are chosen by the human pilot and/or flight deck automation to follow flight plans approved by the controller.The North component of the wind W_N and the East component of the wind W_E are obtained from the NOAA weather data.

In order to enable the investigation of potential accidents and incidents that can occur in each flight phase, the aircraft operations in the NAS have been separated into 11 major flight regimes. These are: stationary at the gate, pushback, taxi to runway, takeoff, climbout, climb-to-cruise, cruise, initial descent, approach, landing, taxi to gate. The software is designed such that the salient motion characteristics of aircraft in historic accidents and incidents in each of these flight regimes and their impact on NAS operations can be formulated and analyzed.

The motion of the aircraft on the ground and the motion of ground vehicles can be simulated by eliminating the altitude dynamics, and assuming that the flight path angle γ is zero. Moreover, the wind has negligible effect on aircraft motion when it is on the ground. With these simplifications, the equations of motion for the aircraft moving on the ground, and the equations of motion for the ground vehicle, are:

$$\dot{\lambda} = \frac{V\cos\chi}{R_e + h} \tag{2.4}$$

$$\dot{\tau} = \frac{V\sin\chi}{(R_e + h)\cos\lambda} \tag{2.5}$$

The control variables employed by the pilot or the ground vehicle are the speed V and the course angle χ used to move along the ramp, taxiways, and up to the runway. The aircraft and ground vehicle dynamic equations are integrated using the first-order Euler integration method.

Navigation and Flight-Deck Automation Systems

The main navigation aids currently used by commercial aircraft in the NAS are the GPS, the Inertial Navigation System and the Instrument landing System (or Microwave Landing System). Navigational errors can cause the aircraft to deviate from specified flight procedures, potentially leading to unsafe operating conditions. In order to assist in the investigation of the effect of these errors on flight safety, the NATS software allows the user to introduce both deterministic and random errors into the aircraft position and velocity components and in the sequence of operations.

Modern commercial aircraft are operated with the aid of flight deck automation systems. These include the autopilot and the autothrottle settings accessible through the Mode Control Panel, with the Flight Management System providing trajectory tracking, fuel management and other higher-level automation functions. Pilots access the FMS functionality through the control display units.

Simplified representations of these automation systems are provided in the NATS software. For instance, using the aircraft flight plan, the FMS subsystem will generate the course angle required to track the series of latitude-longitude waypoints using the formula [2]:

$$\chi = \tan^{-1}\left(\frac{\sin(\tau_{i+1}-\tau)\cos\lambda_{i+1}}{\sin\lambda_{i+1}\cos\lambda - \cos\lambda_{i+1}\sin\lambda\cos(\tau_{i+1}-\tau)}\right) \tag{2.6}$$

In this expression, (λ, τ) is the current latitude-longitude location of the vehicle, and $(\lambda_{i+1}, \tau_{i+1})$ is the latitude-longitude location of the next waypoint in the flight plan.The altitude changes required by the flight plan are implemented using the BADA data for the specific aircraft type.

The autothrottle function is simulated in NATS by selecting the airspeed from the BADA corresponding to a specific aircraft type and flight regime, and using these to integrate the equations of motion.

Just as in the case of the navigation system, deterministic and random error components or operational errors can be introduced into the automation system outputs to investigate their effects on the system safety.

Next generation aircraft are likely to have additional automation available on the flight deck such as airborne self-separation, and tools for trajectory-based operations. NATS software is being designed to enable the investigation of the potential error sources in these systems, and their impact on the NAS safety.

Communication and Surveillance Systems

Most of the communications between the controller and the pilot in the present air traffic control system are achieved over VHF/UHF radio. The contents of the communications typically involve flight plan modifications, changes in cruise altitudes, speed and heading advisories, and potential weather deviations. NATS provides functions for introducing communication errors and to assess their impact on the NAS safety. Moreover, the effect of the terrain on communications between aircraft and between aircraft and the ground can also be assessed in NATS, as will be discussed in Subsection 2.2.2.

Dependent surveillance of the most of the aircraft in NAS is currently achieved through a network of ground-based tracking radars interacting with Mode C transponders onboard aircraft. FAA has mandated that by January 1, 2020, every aircraft operating in the NAS that are currently required to carry Mode C transponders, must be equipped with ADS-B Out (Automatic Dependent Surveillance-Broadcast) capability. The ADS-B system derives position estimates with the aid of GPS satellites, augments the data with aircraft-derived speed, heading and vertical speed information which is then broadcast. The aircraft state estimates are highly accurate and are available at much higher sample rates than the radar.

However, the system is currently susceptible to jamming, which can significantly degrade their performance. Moreover, the position estimates can contain significant error under certain GPS satellite constellation configuration relative to aircraft. NATS software provides functions for modeling these and other known susceptibilities to assess their impact on the system safety.

2.2.2 Environment Interface

Environmental factors have contributed to several well-known accidents in the NAS. Functions are provided in the NATS for modeling the nominal and off-nominal behavior of the system under the influence of environmental factors such as adverse terrain and weather. Some aspects of these environmental factors are discussed in the following subsections.

Terrain

The terrain over the regions of the NAS places severe operational constraints on flight operations. Firstly, the terrain can pose direct hazards to aviation, by requiring higher navigation precision to achieve safe arrivals and departures at certain airports. Secondly, the terrain can limit the line-of-sight between various regions in the NAS, making it difficult to carry out VHF/UHF communications between controllers and pilots. As an example, PUT FIGURE illustrates the effect of the terrain on line-of-sight communications in the vicinity of the Salt Lake City international airport (KSLC)

2.2.3 Entity Interface

Describe: Equipment Interface. To be completed.

2.2.4 Simulation Interface

Describe: Simulation Interface. To be completed.

2.3 Safety Metrics

Safety metrics To be completed.

Chapter 3

Datasets and Models

The sections in this Chapter are to be completed.

3.1 Aircraft Performance Model

BADA

3.2 National Airspace Data

CIFP / NFDC /sector data

3.3 Airport Surface Model

Airport surface modeling

3.4 Terrain Data and Model

USGS 3DEP and how er are using it in NATS

3.5 Wind Model and Data

NOAA RAP and how it is used in NATS

3.6 Weather Data

NOAA aviation weather data and models

3.7 Aircraft Cost and Load Data

Book value/cost of aircraft and passenger load factor data

Chapter 4

Algorithms

4.1 Aircraft Propagation Algorithms

Here we list the 25 state propagation algorithms.

4.1.1 Nomenclature

λ -Latitude of aircraft.
τ - Longitude of the aircraft.
h - Altitude of the aircraft.
V - Groundspeed of the aircraft (wind subtracted from TAS).
γ - Flight path angle.
χ - Course angle.
Δt - Time step.

4.1.2 Preliminaries

Calculate flight path angle between two points with positions (λ_1, τ_1, h_1) and (λ_2, τ_2, h_2), assume $h_2 \geq h_1$

$$\gamma = \tan^{-1}\left[\frac{h_2 - h_1}{d_{gc}(\lambda_1, \tau_1, h_1, \lambda_2, \tau_2, h_1)}\right] \tag{4.1}$$

Calculate the great circle distance between two positions (λ_1, τ_1, h_1) and (λ_2, τ_2, h_2).

$$d_{gc} = 2\,(R_e + h)\,\arcsin\sqrt{\sin^2\left(\frac{|\lambda_1 - \lambda_2|}{2}\right) + \cos(\lambda_1)\,\cos(\lambda_2)\,\sin^2\left(\frac{|\tau_1 - \tau_2|}{2}\right)}; \tag{4.2}$$

4.1.3 Propagation Algorithms for Each Phase

Algorithm 1 Gate

```
Given χ = χ_gate
if No pushback clearance then
    v = 0, ḣ = 0 and χ = χ_gate
end if
```

Algorithm 2 Pushback

```
Given $L_{pushback}$, $v_{tug}$, $\Delta t$ .
$s = 0$, $v = 0$, $\chi = \chi_{gate}$.
$\dot{h} = 0$, $h = h_{apt}$                          ▷ $h_{apt}$ is the altitude of airport.
while $s < L_{pushback}$ do
    if no clearance received then
        continue
    else
        $v = v_{tug}$
    end if
    if $L_{pushback} \leq s + v\,\Delta t$ then
        $s = L_{pushback}$
        $\chi = \chi_{ramp}$
        break
    else
        $s = s + v\,\Delta t$
        $\chi = \chi_{pushback}$                      ▷ Use waypoint to waypoint tracking here
    end if
end while
```

Algorithm 3 Ramp

Given $L_{ramp} = \sum_{i=1}^{N} L_{ramp_i}$ ▷ Sum of all ramp lengths for N ramp legs.
Given v_{ramp}, Δt .
$s = 0$, $v = 0$
$\dot{h} = 0$, $h = h_{apt}$ ▷ h_{apt} is the altitude of airport.
$\chi = \chi_{ramp_1}$, ▷ χ_{ramp_1} is course of first rampway.
$L_{tg} = L_{ramp}$ ▷ L_{tg} is the length to go.
$k = 1$ ▷ Ramp leg counter.
while $L_{tg} > 0$ **do**
    Calculate L_{ramp_r} for every $r = k, \ldots, N$. ▷ In case there is a perturbation during simulation this takes care of the everything.
    $L_{tg} = \sum_{j=k+1}^{N} L_{ramp_j} + d_{gc}(\lambda_c, \tau_c, h_c, \lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$ ▷ (λ_c, τ_c, h_c) is the current position and $(\lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$ is the next ramp waypoint.
    if no clearance received **then**
        continue
    else
        $v = v_{ramp}$
    end if
    if $L_{tg} \leq v\,\Delta t$ **then**
        $s = L_{ramp}$
        $\chi = \chi_{ramp}$, ▷ χ_{ramp} is the course at the end of ramp.
        break
    else
        $s = s + v\,\Delta t$
        $\chi = \chi_{ramp}(s)$, ▷ $\chi_{ramp}(s)$ is the course at the current ramp position. Use waypoint to waypoint tracking
    end if
    if $L_{tg} - v\,\Delta t \leq \sum_{j=k+1}^{N} L_{ramp_j}$ **then**
        $k = k + 1$
    end if
end while

Algorithm 4 Taxi

Given $L_{taxi} = \sum_{i=1}^{N} L_{taxi_i}$ ▷ Sum of all taxi lengths for N taxi legs.
Given v_{taxi}, Δt .
$s = 0$, $v = 0$
$\dot{h} = 0$, $h = h_{apt}$ ▷ h_{apt} is the altitude of airport
$\chi = \chi_{taxi_1}$, ▷ χ_{taxi_1} is course of first taxiway.
$L_{tg} = L_{taxi}$ ▷ L_{tg} is the length to go.
$k = 1$ ▷ Taxi leg counter.
while $L_{tg} > 0$ **do**
    Calculate L_{taxi_r} for every $r = k, \ldots, N$ ▷ In case there is a perturbation during simulation this takes care of the everything.
    $L_{tg} = \sum_{j=k+1}^{N} L_{taxi_j} + d_{gc}(\lambda_c, \tau_c, h_c, \lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$ ▷ (λ_c, τ_c, h_c) is the current position and $(\lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$ is the next taxi waypoint.
    if no clearance received **then**
        continue
    else
        $v = v_{taxi}$
    end if
    if $L_{tg} \leq v\,\Delta t$ **then**
        $s = L_{taxi}$
        $\chi = \chi_{rwy}$, ▷ χ_{rwy} is runway course
        break
    else
        $s = s + v\,\Delta t$
        $\chi = \chi_{taxi}(s)$, ▷ $\chi_{taxi}(s)$ is the course at the current taxi position. Use waypoint to waypoint tracking.
    end if
    if $L_{tg} - v\,\Delta t \leq \sum_{j=k+1}^{N} L_{taxi_j}$ **then**
        $k = k + 1$
    end if
end while

Algorithm 5 Runway Threshold/ Hold Intersection

if No takeoff clearance **then**
    $v = 0$, $\dot{h} = 0$, $\chi = \chi_{rwy}$.
end if

Algorithm 6 Takeoff and Maintain runway course leg

Given $L_{takeoff}$, v_{v_2}, Δt .
Given $v_{wind} = [v_{dw}, v_{cw}]$ ▷ v_{dw} and v_{cw} are downwind and crosswind, respectively.
Given $h = h_{wp_1}$ ▷ h_{wp_1} is the altitude of the first flight plan point after take off.
$s = 0$, $v_t = 0$
$\dot{h} = 0$, $h = h_{apt}$ ▷ h_{apt} is the altitude of airport
$\chi = \chi_{rwy}$
$a = \frac{(v_{v_2} - v_{dw})^2}{2\, L_{takeoff}}$
$L_{wp} = L_{takeoff}$
while $h < h_{wp_1}$ **do**
  if no takeoff clearance received **then**
    $v = 0, \dot{h} = 0$ $\chi = \chi_{rwy}$
    continue
  end if
  if $\dot{h} = 0$ **then**
    $v = a\, \Delta t$ ▷ Use this rule before wheels off.
  else
    $\dot{h} = \dot{h}_{BADA}(h)$ ▷ Use BADA after takeoff. Linearly interpolate for $\dot{h}$ for a given altitude
    $v_t = \{v_{CIFP}(h), v_{BADA}(h)\}$ ▷ If CIFP speed given use that or use BADA after takeoff. Linearly interpolate for v for a given altitude
    $\gamma = \sin^{-1}\left(\frac{\dot{h}}{v_t}\right)$ ▷ Flight path angle
    $v = v_t \cos\gamma$
  end if
  if $L_{wp} \leq s + \frac{1}{2}\, a\, \Delta t^2$ **and** $\dot{h} = 0$ **then**
    $\dot{h} = \dot{h}_{BADA}(h)$ ▷ Get initial $\dot{h}$ from BADA
    $L_{wp} = L_{wp} + d_{gc}(\lambda_{v_2}, \tau_{v_2}, h_{v_2}, \lambda_{wp_1}, \tau_{wp_1}, h_{wp_1})$ ▷ $d_{gc}(\cdot)$ is the GC distance between v_2 point and the next waypoint in flight plan.
    $v_t = v_{v_2}$ ▷ Set speed as the v_2 speed if $L_{takeoff}$ reached between two time intervals.
    $\gamma = \sin^{-1}\left(\frac{\dot{h}}{v_t}\right)$ ▷ Flight path angle
    $v = v_t \cos\gamma$
    $s = L_{wp} + v_{v_2}\left(\Delta t - \sqrt{\frac{2(L_{wp} - s)}{a}}\right)$ ▷ Set $s = L_{wp}$ the incremental distance flown if $L_{takeoff}$ reached between two time intervals.
    $h = h + \dot{h}\left(\Delta t - \sqrt{\frac{2(L_{wp} - s)}{a}}\right)$
    continue
  else if $L_{wp} \leq s + v\, \Delta t$ **and** $\dot{h} \neq 0$ **then**
    $h = h_{wp_1}$, $s = L_{wp}$
    break
  else
    if $\dot{h} = 0$ **then**
      $s = s + \frac{1}{2} a\, \Delta t^2$
    else
      $s = s + v\, \Delta t$
    end if
  end if
  $h = h + \dot{h}\Delta t$
end while

Algorithm 7 Climb out

Given h_{TRACON}, $h_{max_{BADA}}$ ▷ Altitude of the TRACON for departing airport and the maximum altitude in BADA table less than h_{TRACON}, respectively
Given Δt ,N ▷ Number of SID legs before h_{TRACON}.
Given $L_{leg} = [L_{leg_1}, \ldots, L_{leg_N}]$, ▷ Length of each SID leg.
Given h_{wp_1} ▷ h_{wp_1} is the altitude of the first flight plan point after take off.
$h = h_{wp_1}$, $\dot{h} = \dot{h}_{BADA}(h_{wp_1})$
$v_t = \{v_{CIFP}(h_{wp_1}), v_{BADA}(h_{wp_1})\}$ ▷ If CIFP speed given use that or use the BADA speed.
$\chi = \chi_{rwy}$ ▷ Set χ to the runway course
$L_{tg} = \sum_{j=1}^{N} L_{leg_j}$ ▷ L_{tg} is the length to go.
$s = 0$ ▷ Distance along the SID
$k = 0$ ▷ Leg counter
while $h < h_{TRACON}$ **do**
    $v_t = \{v_{CIFP}(h), v_{BADA}(h)\}$, ▷ Use Interpolation from BADA. If speed given in CIFP use that for interpolation limits.
    if $h_{max_{BADA}} \leq h < h_{TRACON}$ **then**
        Use leveling off algorithm for $\dot{h}$
    else
        $\dot{h} = \dot{h}_{BADA}(h)$
    end if
    $\gamma = \sin^{-1}\left(\frac{\dot{h}}{v_t}\right)$ ▷ Flight path angle
    $v = v_t \cos\gamma$
    if $L_{tg} \leq v\,\Delta t$ **then**
        $s = \sum_{j=1}^{N} L_{leg_j}$
        $h = h_{TRACON}$,
        $\chi = \chi_{leg_N}$ ▷ χ_{leg_N} is the course of the last leg before h_{TRACON}
        break
    end if
    Calculate L_{leg_r} for every $r = k, \ldots, N$ ▷ In case there is a perturbation during simulation this takes care of the everything.
    $L_{tg} = \sum_{j=k+1}^{N} L_{leg_j} + d_{gc}(\lambda_c, \tau_c, h_c, \lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$ ▷ (λ_c, τ_c, h_c) is the current position and $(\lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$ is the next flight plan waypoint.
    $h = h + \dot{h}\,\Delta t$
    $s = s + v\,\Delta t$
    if $L_{tg} - v\,\Delta t \leq \sum_{i=k+1}^{N} L_{leg_i}$ **then**
        $k = k + 1$
    end if
    $\chi = \chi_{leg_k}$ ▷ χ_{leg_k} is the course of the k^{th} leg
end while

Algorithm 8 Hold in pattern after climb out

```
Given h_TRACON                          ▷ Altitude of the TRACON for departing airport
Given L_H                               ▷ Length of the holding leg
Given χ_H                               ▷ Course of the holding leg
Given Δt
v = v_BADA(h_TRACON), ḣ = 0
s = 0, h = h_TRACON
while no clearance received do
    χ = χ_H
    if s + vΔt > L_H then
        s = s + vΔt − L_H
    else
        s = s + vΔt
    end if
end while
```

Algorithm 9 Climb to Cruise

Given h_{TRACON}, h_{cruise} ▷ Altitude of the TRACON for departing airport and cruise altitude, respectively
Given Δt ,N ▷ Number of SID legs after h_{TRACON}
Given $L_{leg} = [L_{leg_1}, \ldots, L_{leg_N}]$, ▷ Length of each SID Leg.
Given h_{wp_1} ▷ h_{wp_1} is the altitude of the first flight plan point after h_{TRACON}.
$h = h_{wp_1}$, $\dot{h} = \dot{h}_{BADA}(h_{wp_1})$
$v_t = \{v_{CIFP}(h_{wp_1}), v_{BADA}(h_{wp_1})\}$ ▷ If CIFP speed given use that or use the BADA speed.
$\chi = \chi_{leg_1}$ ▷ Set course to first leg of SID after h_{TRACON}.
$L_{tg} = \sum_{j=1}^{N} L_{leg_j}$ ▷ L_{tg} is the length to go.
$s = 0$ ▷ Distance along the SID
$k = 0$ ▷ Leg counter
while $L_{tg} > 0$ **and** $h < h_{cruise}$ **do**
  if no clearance received **then**
    Go to Algorithm 8
    continue
  end if
  $v_t = \{v_{CIFP}(h), v_{BADA}(h)\}$, ▷ Use Interpolation from BADA. If speed limit given in CIFP use that.
  if $h_{max_{BADA}} \leq h < h_{cruise}$ **then** ▷ $h_{max_{BADA}}$ is the maximum altitude in BADA table less than h_{cruise}
    Use leveling off algorithm for $\dot{h}$
  else
    $\dot{h} = \dot{h}_{BADA}(h)$ ▷ Use Interpolation from BADA.
  end if
  $\gamma = \sin^{-1}\left(\frac{\dot{h}}{v_t}\right)$ ▷ Flight path angle
  $v = v_t \cos\gamma$
  if $L_{tg} \leq v\,\Delta t$ **then**
    $s = \sum_{j=1}^{N} L_{leg_j}$
    $h = h_{cruise}$
    $\chi = \chi_{leg_N}$ ▷ Course of last leg before cruise.
    break
  end if
  Calculate L_{leg_r} for every $r = k, \ldots, N$ ▷ In case there is a perturbation during simulation this takes care of the everything.
  $L_{tg} = \sum_{j=k+1}^{N} L_{leg_j} + d_{gc}(\lambda_c, \tau_c, h_c, \lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$ ▷ (λ_c, τ_c, h_c) is the current position and $(\lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$ is the next flight plan waypoint.
  $s = s + v\,\Delta t$
  $h = h + \dot{h}\,\Delta t$
  if $L_{tg} - v\,\Delta t \leq \sum_{i=k+1}^{N} L_{leg_i}$ **then**
    $k = k + 1$
  end if
  $\chi = \chi_{leg_k}$ ▷ χ_{leg_k} is the course of the k^{th} leg
end while

Algorithm 10 Top of climb

```
Given h_cruise, v_cruise                                  ▷ Cruise altitude and speed, respectively
Given (λ_TOC, τ_TOC)                                      ▷ Latitude and longitude of TOC
if h = h_cruise then
    ḣ = 0
    v = v_cruise
    χ = χ_gc(λ_TOC, τ_TOC, h_cruise, λ_wp1, τ_wp1, h_cruise)   ▷ (λ_wp1, τ_wp1) are the latitude and longitude of the first cruise waypoint
end if
```

Algorithm 11 Cruise

Given h_{cruise}, v_{cruise} ▷ Cruise altitude and speed, respectively
Given N ▷ Number of cruise legs
Given $L_{leg} = [L_{leg_1}, \ldots, L_{leg_N}]$, ▷ Length of each cruise leg.
Given Δt
$h = h_{cruise}$, $v = v_{cruise}$, $\chi = \chi_{leg_1}$, $\dot{h} = 0$ ▷ Initial states
$s = 0$, ▷ Distance and leg counter
Set $L = \sum_{k=1}^{N-1} L_{leg_k}$
Set $L_{tg} = L$ ▷ L_{tg} is the length to go.
$k = 0$ ▷ Leg counter
while $L_{tg} > 0$ **do**
  if $L_{tg} \leq v\,\Delta t$ **then**
    if no clearance received **then** ▷ Maintain course and continue moving using the same speed and altitude
      Execute Algorithm 12
      continue
    else
      $s = L$
      $\chi = \chi_{leg_{N-1}}$
      $h = h_{cruise}$
      break
    end if
  end if
  Calculate L_{leg_r} for every $r = k, \ldots, N$ ▷ In case there is a perturbation during simulation this takes care of the everything.
  $L_{tg} = \sum_{j=k+1}^{N} L_{leg_j} + d_{gc}(\lambda_c, \tau_c, h_c, \lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$ ▷ (λ_c, τ_c, h_c) is the current position and $(\lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$ is the next flight plan waypoint.
  $s = s + v\Delta t$
  if $L_{tg} - v\,\Delta t \leq \sum_{i=k+1}^{N} L_{leg_i}$ **then**
    $k = k + 1$
  end if
  $\chi = \chi_{leg_k}$ ▷ χ_{leg_k} is the course of the k^{th} leg
end while

Algorithm 12 Hold in pattern enroute

Given h_{cruise}, v_{cruise} ▷ Cruise altitude and speed
Given L_H ▷ Length of the holding leg
Given χ_H ▷ Course of the holding leg
Given Δt
$v = v_{cruise}$, $h = h_{cruise}$, $\dot{h} = 0$
$s = 0$, $h = h_{cruise}$
while no clearance received **do**
  $\chi = \chi_H$
  if $s + v\Delta t > L_H$ **then**
    $s = s + v\Delta t - L_H$
  else
    $s = s + v\Delta t$
  end if
end while

Algorithm 13 Top of descent

Given h_{cruise}, v_{cruise} ▷ Cruise altitude and speed, respectively
Given $(\lambda_{TOD}, \tau_{TOD})$ ▷ Latitude and longitude of TOD
Given χ_{leg_1} ▷ χ_{leg_1} is the course along first STAR leg.
Given Δt
if $h = h_{cruise}$, $\lambda = \lambda_{TOD}$, $\tau = \tau_{TOD}$ **then**
  $\dot{h} = \dot{h}_{BADA}(h_{cruise})$
  $\chi = \chi_{leg_1}$
  $v_t = v_{BADA}(h_{cruise})$
  $\gamma = \sin^{-1}\left(\frac{\dot{h}}{v_t}\right)$ ▷ Flight path angle
  $v = v_t \cos\gamma$
end if

Algorithm 14 Initial Descent

Given h_{TRACON}, h_{cruise} ▷ Altitude of the TRACON for arriving airport and cruise altitude, respectively
Given $h_{min_{BADA}}$ ▷ $h_{min_{BADA}}$ is the minimum altitude in BADA table greater than h_{TRACON}
Given h_{CA} ▷ Minimum altitude of the Class A airspace $h_{CA} > h_{TRACON}$
Given Δt ,N ▷ Number of STAR legs before h_{TRACON}
Given $L_{leg} = [L_{leg_1}, \ldots, L_{leg_N}]$, ▷ Length of each STAR Leg.
$h = h_{cruise}$, $v_t = v_{cruise}$, $\chi = \chi_{leg_1}$, $\dot{h} = 0$ ▷ Initial states before descent
$L_{tg} = \sum_{j=1}^{N} L_{leg_j}$ ▷ L_{tg} is the length to go.
$s = 0$ ▷ Distance along the STAR
$k = 0$ ▷ Leg counter
while $L_{tg} > 0$ **and** $h > h_{TRACON}$ **do**
  $v_t = \{v_{CIFP}(h), v_{BADA}(h)\}$, ▷ Use Interpolation from BADA. If speed limit given in CIFP use that.
  if $h_{TRACON} < h \leq h_{CA}$ **then**
    if no clearance received **then**
      Execute Algorithm 18.
    end if
  end if
  if $h_{TRACON} < h \leq h_{min_{BADA}}$ **then**
    Use leveling off algorithm for $\dot{h}$
  else
    $\dot{h} = \dot{h}_{BADA}(h)$ ▷ Use Interpolation from BADA.
  end if
  $\gamma = \sin^{-1}\left(\frac{\dot{h}}{v_t}\right)$ ▷ Flight path angle
  $v = v_t \cos\gamma$
  if $L_{tg} \leq v\,\Delta t$ **then**
    $s = \sum_{j=1}^{N} L_{leg_j}$
    $h = h_{TRACON}$
    $\chi = \chi_{leg_N}$
    break
  end if
  Calculate L_{leg_r} for every $r = k, \ldots, N$ ▷ In case there is a perturbation during simulation this takes care of the everything.
  $L_{tg} = \sum_{j=k+1}^{N} L_{leg_j} + d_{gc}(\lambda_c, \tau_c, h_c, \lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$ ▷ (λ_c, τ_c, h_c) is the current position and $(\lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$ is the next flight plan waypoint.
  $d_{leg} = d_{gc}(\lambda_c, \tau_c, h_c, \lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$, $\theta = \frac{d_{leg}}{R_e + h}$ ▷ R_e is the radius of the earth.
  $\dot{s} = \theta\, v_t \sin\gamma + v_t \cos\gamma$
  $s = s + \dot{s}\,\Delta t$
  $h = h + \dot{h}\,\Delta t$
  if $L_{tg} - v\,\Delta t \leq \sum_{i=k+1}^{N} L_{leg_i}$ **then**
    $k = k + 1$
  end if
  $\chi = \chi_{leg_k}$ ▷ χ_{leg_k} is the course of the k^{th} leg
end while

Algorithm 15 Final Descent

Given h_{TRACON}, h_{IAF} ▷ Altitude of the TRACON for arriving airport and altitude of initial approach fix, respectively

Given Δt ,N ▷ Number of STAR legs before h_{IAF} and after h_{TRACON}

Given $L_{leg} = [L_{leg_1}, \dots, L_{leg_N}]$, ▷ Length of each STAR Leg.

$h = h_{TRACON}$, ▷ Initial altitude before final descent

$v_t = \{v_{CIFP}(h_{TRACON}), v_{BADA}(h_{TRACON})\}$, ▷ If CIFP speed given use that or use the BADA speed.

$\dot{h} = \dot{h}_{BADA}(h)$, $\gamma = \sin^{-1} \frac{\dot{h}}{v_t}$, $v = v_t \cos\gamma$

$L_{tg} = \sum_{j=1}^{N} L_{leg_j}$ ▷ L_{tg} is the length to go

$s = 0$ ▷ Distance along the STAR

$k = 0$ ▷ Leg counter

while $L_{tg} > 0$ **and** $h > h_{IAF}$ **do**

  $v_t = \{v_{CIFP}(h), v_{BADA}(h)\}$, ▷ Use Interpolation from BADA. If speed limit given in CIFP use that.

  $\gamma = \gamma_{leg_k}$ ▷ Calculate flight path angle for k^{th} leg (See in eqn. 4.1)

  $\dot{h} = v_t \sin(\gamma)$ ▷ Calculate $\dot{h}$ from FPA.

  $v = v_t \cos\gamma$

  if $L_{tg} \leq v\,\Delta t$ **then**

    $s = \sum_{j=1}^{N} L_{leg_j}$

    $h = h_{IAF}$

    $\chi = \chi_{leg_N}$

    break

  end if

  Calculate L_{leg_r} for every $r = k, \dots, N$ ▷ In case there is a perturbation during simulation this takes care of the everything.

  $L_{tg} = \sum_{j=k+1}^{N} L_{leg_j} + d_{gc}(\lambda_c, \tau_c, h_c, \lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$ ▷ (λ_c, τ_c, h_c) is the current position and $(\lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$ is the next flight plan waypoint.

  $d_{leg} = d_{gc}(\lambda_c, \tau_c, h_c, \lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$, $\theta = \frac{d_{leg}}{R_e + h}$ ▷ R_e is the radius of the earth.

  $\dot{s} = \theta\, v_t \sin\gamma + v_t \cos\gamma$

  $s = s + \dot{s}\,\Delta t$

  $h = h + \dot{h}\,\Delta t$

  if $L_{tg} - v\,\Delta t \leq \sum_{i=k+1}^{N} L_{leg_i}$ **then**

    $k = k + 1$

  end if

  $\chi = \chi_{leg_k}$ ▷ χ_{leg_k} is the course of the k^{th} leg

end while

Algorithm 16 Approach

Given h_{IAF}, h_{FAF}, h_{rtp} $\triangleright$ Altitude of the of initial approach fix, final approach fix and runway touchdown point respectively
Given v_{TD} $\triangleright$ Touchdown speed of the aircraft.
Given Δt ,N $\triangleright$ Final approach speed and altitude restriction points
Given $L_{leg} = [L_{leg_1}, \ldots, L_{leg_N}]$, $\triangleright$ Length of each approach legs.
$h = h_{IAF}$ $\triangleright$ Initial altitude before approach
$v_t = \{v_{CIFP}(h_{IAF}), v_{BADA}(h_{IAF})\}$, $\triangleright$ If CIFP speed given use that or use the BADA speed.
$\dot{h} = \dot{h}_\gamma(h)$, $v = v_t \cos\gamma$ $\triangleright$ $\dot{h}_\gamma(h)$ is the $\dot{h}$ at the flight path angle from previous phase. Similar arguments hold for v.
$L_{tg} = \sum_{j=1}^{N} L_{leg_j}$ $\triangleright$ L_{tg} is the length to go
$s = 0$ $\triangleright$ Distance along the approach segment.
$k = 0$ $\triangleright$ Leg counter
while $L_{tg} > 0$ **and** $h > h_{rwy}$ **do**
    $v_t = \{v_{CIFP}(h), v_{BADA}(h)\}$, $\triangleright$ Use Interpolation from BADA. If speed limit given in CIFP use that.
    $\gamma = \gamma_{leg_k}$ $\triangleright$ Calculate flight path angle for k^{th} leg (See in eqn. 4.1)
    $\dot{h} = v_t \sin(\gamma)$ $\triangleright$ Calculate $\dot{h}$ from FPA.
    $v = v_t \cos\gamma$
    if missed approach **then**
        Go to Algorithm 17 and then follow Algorithm 18
    end if
    if $h_{rtp} < h \leq h_{FAF}$ **then**
        Use leveling off algorithm for $\dot{h}$.
    end if
    if $L_{tg} \leq v\, \Delta t$ **then**
        $h = h_{rtp}$, $\dot{h} = 0$
        $s = \sum_{j=1}^{N} L_{leg_j}$
        $\chi = \chi_{rwy}$
        $v = v_{TD}$
        break
    end if
    Calculate L_{leg_r} for every $r = k, \ldots, N$ $\triangleright$ In case there is a perturbation during simulation this takes care of the everything.
    $L_{tg} = \sum_{j=k+1}^{N} L_{leg_j} + d_{gc}(\lambda_c, \tau_c, h_c, \lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$ $\triangleright$ (λ_c, τ_c, h_c) is the current position and $(\lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$ is the next flight plan waypoint
    $d_{leg} = d_{gc}(\lambda_c, \tau_c, h_c, \lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$, $\theta = \frac{d_{leg}}{R_e + h}$ $\triangleright$ R_e is the radius of the earth.
    $\dot{s} = \theta\, v_t \sin\gamma + v_t \cos\gamma$
    $s = s + \dot{s}\, \Delta t$
    if $L_{tg} - v\, \Delta t \leq \sum_{i=k+1}^{N} L_{leg_i}$ **then**
        $h = h + \dot{h}\, \Delta t$
        $k = k + 1$
    end if
    $\chi = \chi_{leg_k}$ $\triangleright$ χ_{leg_k} is the course of the k^{th} leg
end while

Algorithm 17 Go around for missed approach

Given h_{MA} ▷ Altitude where missed approach was determined.
Given h_{HP} ▷ Altitude where the aircraft will hold $h_{HP} \geq h_{MA}$
Given v_{MA} ▷ Speed at missed approach.
Given $\chi_{leg_{GA}}$ ▷ Course along the go around leg
Given L_{GA} ▷ Length of the go around leg till the hold pattern.
Given Δt
$v = v_{MA}$, $\dot{h} = 0$, $\chi = \chi_{leg_{GA}}$.
$s = 0$, $h = h_{MA}$.
while $s < L_{GA}$ **and** $h < h_{HP}$ **do**
    $\dot{h} = \dot{h}_{BADA}$ ▷ Use climb rates here
    $v_t = \{v_{CIFP}(h), v_{BADA}(h)\}$, ▷ Use Interpolation from BADA. If speed limit given in CIFP use that.
    $\gamma = \sin^{-1}\left(\frac{\dot{h}}{v_t}\right)$ ▷ Flight path angle
    $v = v_t \cos\gamma$
    $\chi = \chi_{leg_{GA}}$
    $s = s + v\Delta t$
    $h = h + \dot{h}\Delta t$
end while

Algorithm 18 Hold in descent

Given h_H, v_H ▷ Altitude and speed of the holding pattern
Given χ_H ▷ Course of the holding pattern leg
Given Δt
$\dot{h} = 0$
$h = h_H$, $s = 0$
while no clearance received **do**
    $v = \{v_{CIFP}(h), v_H\}$, ▷ Use Interpolation from BADA. If speed limit given in CIFP use that.
    $\chi = \chi_H$
    if $s + v\Delta t > L_H$ **then**
        $s = s + v\Delta t - L_H$
    else
        $s = s + v\Delta t$
    end if
end while

Algorithm 19 Touchdown

Given v_{td}, χ_{rwy} ▷ Given touchdown speed and runway course.
$v = v_{td}$
$\dot{h} = 0$
$\chi = \chi_{rwy}$
$h = h_{rtp}$ ▷ h_{rtp} is the altitude of runway touchdown point (may be same as airport altitude)

Algorithm 20 Landing to stop

Given v_{td}, χ_{rwy} ▷ Given touchdown speed and runway course.
Given $L_{landing}$ ▷ Landing length
Given $v_{wind} = [v_{dw}, v_{cw}]$ ▷ v_{dw} and v_{cw} are downwind and crosswind, respectively.
$\dot{h} = 0$,$h = h_{apt}$ ▷ h_{apt} is the altitude of airport
$\chi = \chi_{rwy}$
$a = \frac{(v_{td} - v_{dw})^2}{2\, L_{landing}}$
$s = 0$, $v = v_{td} - v_{dw}$
while $s < L_{landing}$ **do**
    $s = s + \frac{1}{2}\, a \Delta t^2$
end while

Algorithm 21 Runway exit

Given χ_{taxi} ▷ Given course of first taxiway
Given v_{taxi} ▷ Speed of taxi.
Given L_{re} ▷ Length of the arc needed for transition from runway to taxiway.
$\dot{h} = 0$, $\chi = \chi_{rwy}$, $v = v_{taxi}$ ▷ Initial conditions
$s = 0$, $h = h_{apt}$ ▷ h_{apt} is the altitude of airport
while $\chi \neq \chi_{taxi}$ **and** $s < L_{re}$ **do**
    if $s + v\Delta t > L_{re}$ **then**
        $s = L_{re}$
        $\chi = \chi_{taxi}$
        break
    end if
    $\chi = \chi_{rwy} + \frac{\chi_{taxi} - \chi_{rwy}}{L_{re}} s$
    $s = s + v\Delta t$
end while

Algorithm 22 Taxi

Given $L_{taxi} = \sum_{i=1}^{N} L_{taxi_i}$ ▷ Sum of all taxi lengths for N taxi legs.
Given v_{taxi}, Δt .
$s = 0$, $v = 0$
$\dot{h} = 0$, $h = h_{apt}$ ▷ h_{apt} is the altitude of airport
$\chi = \chi_{taxi_1}$, ▷ χ_{taxi_1} is course of first taxiway.
$L_{tg} = L_{taxi}$ ▷ L_{tg} is the length to go.
$k = 1$ ▷ Taxi leg counter.
while $L_{tg} > 0$ **do**
  Calculate L_{taxi_r} for every $r = k, \ldots, N$ ▷ In case there is a perturbation during simulation this takes care of the everything.
  $L_{tg} = \sum_{j=k+1}^{N} L_{taxi_j} + d_{gc}(\lambda_c, \tau_c, h_c, \lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$ ▷ (λ_c, τ_c, h_c) is the current position and $(\lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$ is the next taxi waypoint.
  if no clearance received **then**
    continue
  else
    $v = v_{taxi}$
  end if
  if $L_{tg} \leq v\, \Delta t$ **then**
    $s = L_{taxi}$
    $\chi = \chi_{ramp_1}$, ▷ χ_{rwy} is runway course
    break
  else
    $s = s + v\, \Delta t$
    $\chi = \chi_{taxi}(s)$, ▷ $\chi_{taxi}(s)$ is the course at the current taxi position. Use waypoint to waypoint tracking.
  end if
  if $L_{tg} - v\, \Delta t \leq \sum_{j=k+1}^{N} L_{taxi_j}$ **then**
    $k = k + 1$
  end if
end while

Algorithm 23 Runway Threshold/ Hold Intersection

if No clearance **then**
  $v = 0$, $\dot{h} = 0$ $\chi = \chi_{taxi}$. ▷ χ_{taxi} is the course of previous taxiway.
  $h = h_{apt}$ ▷ h_{apt} is the altitude of airport
end if

Algorithm 24 Ramp

Given v_{ramp}, Δt .
$s = 0$, $v = 0$
$\dot{h} = 0$, $h = h_{apt}$ ▷ h_{apt} is the altitude of airport.
$\chi = \chi_{ramp_1}$, ▷ χ_{ramp_1} is course of first rampway.
$L_{tg} = L_{ramp}$ ▷ L_{tg} is the length to go.
$k = 1$ ▷ Ramp leg counter.
while $L_{tg} > 0$ **do**
    Calculate L_{ramp_r} for every $r = k, \ldots, N$. ▷ In case there is a perturbation during simulation this takes care of the everything.
    $L_{tg} = \sum_{j=k+1}^{N} L_{ramp_j} + d_{gc}(\lambda_c, \tau_c, h_c, \lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$ ▷ (λ_c, τ_c, h_c) is the current position and $(\lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$ is the next ramp waypoint.
    if no clearance received **then**
        continue
    else
        $v = v_{ramp}$
    end if
    if $L_{tg} \leq v\,\Delta t$ **then**
        $s = L_{ramp}$
        $\chi = \chi_{gate}$, ▷ χ_{gate} is the course at of the gate.
    else
        $s = s + v\,\Delta t$
        $\chi = \chi_{ramp}(s)$, ▷ $\chi_{ramp}(s)$ is the course at the current ramp position. Use waypoint to waypoint tracking
    end if
    if $L_{tg} - v\,\Delta t \leq \sum_{j=k+1}^{N} L_{ramp_j}$ **then**
        $k = k + 1$
    end if
end while

Algorithm 25 Gate

Given $\chi = \chi_{gate}$
$v = 0$, $\dot{h} = 0$ and $\chi = \chi_{gate}$
$h = h_{apt}$ ▷ h_{apt} is the altitude of airport

4.2 Taxiway Generation

Taxiway from gate to runway threshold and vice versa. To be completed.

4.3 Controller and Pilot Action

All the human factors stuff such as partial action/ absence /wrong action. To be completed.

4.4 Conflict Detection and Resolution

Everything here given in terms of NED coordinates as the course angle always specified w.r.t true North.

4.4.1 Nomenclature

λ -Latitude of aircraft.
τ - Longitude of the aircraft.
h - Altitude of the aircraft.
V - Groundspeed of the aircraft (wind subtracted from TAS).
γ - Flight path angle.
χ - Course angle.
Δt - Time step.

4.4.2 CDNR Algorithms

Algorithm 26 Geodetic to ECEF

procedure GeodeticToECEFConversion(λ, τ, h, V, γ, χ, R_e)
 $\dot{x}_{ecef} = V \sin\sin\lambda\cos\tau\cos\gamma\cos\chi - V\sin\tau\cos\gamma\sin\chi + V\cos\lambda\cos\tau\sin\gamma$
 $\dot{y}_{ecef} = V\sin\lambda\sin\tau\cos\gamma\cos\chi + V\cos\tau\cos\gamma\sin\chi + V\cos\lambda\sin\tau\sin\gamma$
 $\dot{z}_{ecef} = -V\cos\lambda\cos\gamma\cos\chi + V\sin\lambda\sin\gamma$
 $x_{ecef} = (R_e + h)\cos\lambda\cos\tau$
 $y_{ecef} = (R_e + h)\cos\lambda\sin\tau$
 $z_{ecef} = (R_e + h)\sin\lambda$
 return x_{ecef}, y_{ecef}, z_{ecef}, $\dot{x}_{ecef}$, $\dot{y}_{ecef}$, $\dot{z}_{ecef}$
end procedure

Algorithm 27 Get A and B

procedure GetAandBCoefficients(x, y, z, $\dot{x}$, $\dot{y}$, $\dot{z}$)
 $D = x\dot{y} - y\dot{x}$
 if $D = 0$ **then**
  Denominator is 0. Exiting
  return 0, 0
 end if
 $A = \frac{\dot{y}z - y\dot{z}}{D}$
 $B = \frac{\dot{z}x - z\dot{x}}{D}$
 return A, B
end procedure

Algorithm 28 Get Conflict Point

procedure GetConflictPoint(A_1, B_1, A_2, B_2, h, R_e)
  $z_c = (R_e + h)\sqrt{\frac{1}{1+\left(\frac{A_1-A_2}{A_1B_2-A_2B_1}\right)^2+\left(\frac{B_2-B_1}{A_1B_2-A_2B_1}\right)^2}}$
  $y_c = \frac{A_1-A_2}{A_1B_2-A_2B_1}z_c$
  $x_c = \frac{B_2-B_1}{A_1B_2-A_2B_1}z_c$
  return x_c, y_c, z_c
end procedure

Algorithm 29 Get Time To Intersection

procedure GetTimeToIntersectionPoint(x, y, z, x_c, y_c, z_c, V, h, R_e)
  $\sigma = \cos^{-1}\frac{xx_c+yy_c+zz_c}{(R_e+h)^2}$
  $d_c = (R_e + h)\sigma$
  $t_c = \frac{d_c}{V}$
  return t_c
end procedure

Algorithm 30 Detect Conflict to Go

procedure DetectConflictToGo(λ_1, τ_1, h_1, V_1, γ_1, χ_1, λ_2, τ_2, h_2, V_2, γ_2, χ_2, R_e)
  $h = \frac{h_1+h_2}{2}$ ▷ Placeholder.
  x_1, y_1, z_1, $\dot{x}_1$, $\dot{y}_1$, $\dot{z}_1$ = GeodeticToECEFConversion(λ_1, τ_1, h_1, V_1, γ_1, χ_1, R_e)
  x_2, y_2, z_2, $\dot{x}_2$, $\dot{y}_2$, $\dot{z}_2$ = GeodeticToECEFConversion(λ_2, τ_2, h_2, V_2, γ_2, χ_2, R_e)
  A_1, B_1 = GetAandBCoefficients(x_1, y_1, z_1, $\dot{x}_1$, $\dot{y}_1$, $\dot{z}_1$)
  A_2, B_2 = GetAandBCoefficients(x_2, y_2, z_2, $\dot{x}_2$, $\dot{y}_2$, $\dot{z}_2$)
  x_c, y_c, z_c = GetConflictPoint(A_1, B_1, A_2, B_2, h, R_e)
  t_{c1} = GetTimeToIntersectionPoint(x_1, y_1, z_1, x_c, y_c, z_c, V_1, h, R_e)
  t_{c2} = GetTimeToIntersectionPoint(x_2, y_2, z_2, x_c, y_c, z_c, V_2, h, R_e)
  return True ,t_{c2},t_{c1}
end procedure

Algorithm 31 Conflict Detection

procedure CONFLICTDETECTION(λ_1, τ_1, h_1, V_1, γ_1, χ_1, λ_2, τ_2, h_2, V_2, γ_2, χ_2, R_e, d_{detect})
  if $h_1 < 29000ft$ and $h_2 < 29000ft$ **then**
    if $\|h_1 - h_2\| > 1000$ **then**
      return False,-1,-1
    end if
  else if $h_1 < 29000ft$ and $h_2 \geq 29000ft$ or $h_1 \geq 29000ft$ and $h_2 < 29000ft$ **then**
    if $\|h_1 - h_2\| > 1000$ **then**
      return False,-1,-1
    end if
  else
    if $\|h_1 - h_2\| > 2000$ **then**
      return False,-1,-1
    end if
  end if
  $h = \frac{h_1+h_2}{2}$ ▷ Placeholder.
  $d_gc = d_{gc}(\lambda_1, \tau_1, h, \lambda_2, \tau_2, h)$ ▷ Great circle distance
  if $d_{gc} > d_{initiate}$ **then**
    return False,-1,-1
  end if
  return DetectConflictToGo(λ_1, τ_1, h_1, V_1, γ_1, χ_1, λ_2, τ_2, h_2, V_2, γ_2, χ_2, R_e)
end procedure

Algorithm 32 Number of time steps to hold

procedure FINDNUMBEROFTIMESTEPSTOHOLD(d_l,d_h,V,Δt)
  $\Delta D = d_h - d_l$
  $t = \Delta D/V$
  $n_s = t/\Delta t$
  $n_s = floor(ns) + 1$
  return n_s
end procedure

Algorithm 33 Conflict Resolution

```
procedure ConflictResolution(V_1, V_2, R_e, t_c1, t_c2, d_separation)
    if t_c1 > t_c2 then
        if (t_c1 − t_c2)V_1 < d_separation then
            return 1, FindNumberofTimeStepsToHold( (t_c1 − t_c2)V_1, d_separation, V_1, Δt)
        end if
    else if t_c1 < t_c2 then
        if (t_c2 − t_c1)V_2 < d_separation then
            return 2, FindNumberofTimeStepsToHold( (t_c2 − t_c1)V_2, d_separation, V_2, Δt)
        end if
    else
        t_1 = (t_c1 V_1 − d_separation) / V_1
        t_2 = (t_c2 V_2 − d_separation) / V_2
        if t_1 > t_2 then
            return 1, floor(t_2/Δt) + 1
        else if t_1 < t_2 then
            return 2, floor(t_1/Δt) + 1
        else
            return 1, 1.
        end if
    end if
end procedure
```

Algorithm 34 CDNR

```
procedure ConflictDetectionAndResolution(λ_1, τ_1, h_1, V_1, γ_1, χ_1, λ_2, τ_2, h_2, V_2, γ_2, χ_2, R_e, d_initiate, d_separation)
    IsConflict,t_c1,t_c2 = ConflictDetection(λ_1, τ_1, h_1, V_1, γ_1, χ_1, λ_2, τ_2, h_2, V_2, γ_2, χ_2, R_e, d_initiate)
    if IsConflict then
        d_ac,d_step = ConflictResolution(V_1, V_2, R_e, t_c1, t_c2, d_separation)
        return d_ac,d_step
    else
        Do nothing no conflict
        return -1,-1
    end if
end procedure
```

Algorithm 35 Propagation

Given N- number of aircraft, t_0, t_f and Δt - Propagation time parameters.
procedure PROPAGATION(N,t_0, t_f, Δt)
    $t = t_0$
    while $t < t_f$ **do**
        for $n = 1 \mapsto N$ **do**
            Propagate aircraft n
        end for
        for $n = 2 \mapsto N$ **do**
            for $m = 1 \mapsto n - 1$ **do**
                d_{ac},d_{step} = ConflictDetectionAndResolution(λ_n, τ_n, h_n, V_n, γ_n, χ_n, λ_m, τ_m, h_m, V_m, γ_m, χ_m, R_e, $d_{initiate}$, $d_{separation}$)
                if $d_{ac} == 1$ **then**
                    Delay AC n for d_{step} time intervals.
                    Skip CDNR for pair (m, n) for next d_{step} time intervals.
                else if $d_{ac} == 2$ **then**
                    Delay AC m for d_{step} time intervals.
                    Skip CDNR for pair (m, n) for next d_{step} time intervals.
                else
                    continue
                end if
            end for
        end for
    end while
end procedure

4.5 Weather Avoidance

Weather avoidance algorithm. To be completed.

4.6 Merging and Spacing

This section provides the merging and spacing algorithm that has been implemented in NATS.

Algorithm 36 GetMeterFixPoints

procedure GETMETERFIXPOINTS($aplist[N]$,$starlist[N, M]$)
  $meterFixMap := ap \mapsto fixlist$
  for $n = 1 \mapsto N$ **do**
    $meterFixForAp =$ ▷ Vector of all meterfixes for the airport.
    for $m = 1 \mapsto M$ **do**
      $star = starlist[n, m]$
      $wp = getString(star)$ ▷ Get all strings preceding the number in star
      if $wp \in getWaypoints(star)$ **then** ▷ Find if the waypoint wp in list of waypoints in star.
        $meterFixForAp = [meterFixForAp, wp]$ ▷ Append the wp to meter fix list.
      end if
    end for
    $meterFixMap := aplist[n] \mapsto meterFixForAp$ ▷ Insert in the meterfix map.
  end for
  return $meterFixMap$.
end procedure

Algorithm 37 CreateMeterfixToAcAndDistMap

procedure CREATEMETERFIXTOACANDDISTMAP(ac,$destAp$,aar,$acTargWp$,$meterFixMap$, $\lambda_{targwp}, \tau_{targwp}, h_{targwp}, \lambda_{curr}, \tau_{curr}, h_{curr}$, $meterfixtoAcAndDistMap$)
  $\Delta t_{aar} = \frac{1}{aar}$
  if $acTargWp$ in $meterFixMap[destAp]$ **then**
    $d_{targwp} = d_{gc}(\lambda_{targwp}, \tau_{targwp}, h_{targwp}, \lambda_{curr}, \tau_{curr}, h_{curr})$
    if $\Delta t_{aar} < d_{targwp}/V_{GS}$ **then**
      $dist\_ac\_pair = (ac, d_{targwp})$
      $meterfixtoAcAndDistMap[acTargWp].push\_back(dist\_ac\_pair)$
    end if
  end if
end procedure

Algorithm 38 GetACtoHold

```
procedure GetACtoHold(meterfixtoAcAndDistMap, aclist)
    Given number of aircraft N.
    acToHold = ac ↦ bool                                        ▷ map < string, bool >
    for n = 1 ↦ N do
        acToHold[aclist[n]] = False                ▷ By default we will not hold any aircraft.
    end for
    for iter = meterfixtoAcAndDistMap.begin() ↦ meterfixtoAcAndDistMap.end() do
        meterfixPoint = iter ↦ first
        vectorOfAcAndDist = iter ↦ second
        M = vectorOfAcAndDist.size()
        MinDist = max_numericlimit_double
        ACNotToHold = NONE
        for m = 1 ↦ M do
            ac_dist_pair = vectorOfAcAndDist[m]
            ac = ac_dist_pair.first
            dist = ac_dist_pair.second
            if MinDist > dist then
                MinDist = dist
                ACNotToHold = ac
            end if
        end for
        if ACNotToHold ≠ NONE then
            for m = 1 ↦ M do
                ac_dist_pair = vectorOfAcAndDist[m]
                ac = ac_dist_pair.first
                if ACNotToHold ≠ ac then
                    acToHold[ac] = True
                end if
            end for
        end if
    end for
    return acToHold
end procedure
```

Algorithm 39 Propagation

Given N- number of aircraft, t_0, t_f and Δt - Propagation time parameters.
procedure PROPAGATION(N,t_0, t_f, Δt)
  $t = t_0$
  while $t < t_f$ **do**
    $meterfixtoAcAndDistMap := fixlist \mapsto vector < pair < ac, dist >>$ ▷ Map of all fixes to all aircrafts and their correspoinding distances to the meter fix.
    for $n = 1 \mapsto N$ **do**
      $CreateMeterfixToAcAndDistMap$($aclist[n]$,$destAp$,$aar$,$acTargWp$,$meterFixMap$,$\lambda_{targwp}$, τ_{targwp} $meterfixtoAcAndDistMap$) ▷ Create a map of distance to meterfix and corresponding ac
    end for
    $acToHold = GetACtoHold(meterfixtoAcAndDistMap, aclist)$ ▷ $acToHold = ac \mapsto bool$ gives the list of acs to hold
    for $n = 1 \mapsto N$ **do**
      if actoHold[aclist[n]] == False **then**
        Propagate aircraft n
      end if
      Calculate states at $t + \Delta t$.
    end for
    $t = t + \Delta t$
  end while
end procedure

Algorithm 40 NATS Simulation

procedure PREPROCESSING
  $meterFixMap = GetMeterFixPoints(aplist[N], starlist[N, M])$
end procedure
Propagation($meterFixMap, \ldots$)

Chapter 5

Setting Up and Running NATS

All the sections in this Chapter are to be completed. You can find everything about installation here.

5.1 Dependencies for Running NATS

Prerequisites for running NATS. Find the configuration supported here, and an intro about getting started here.

5.2 Flight Plan

Flight plan has been described here. How to create a flight plan for NATS simulation. Explain the

5.3 The Server Client Structure

Description of the server and the client structure

5.4 Multi-user Interface

Description of the multiuser interface

Chapter 6

Metrics

Discuss the safety metrics that can be calculated from NATS

Chapter 7

Example Use Cases

Provide some example use cases of NATS.

Bibliography

[1] Nuic, A., User manual for the Base of Aircraft Data (BADA) revision 3.11, EEC Technical/Scientific Report No. 13/04/16-01, May, 2013.

[2] Bilimoria, K. D., Sridhar, B., Grabbe, S. R., Chatterji, G. B., and Sheth, K. S., FACET: Future ATM concepts evaluation tool. Air Traffic Control Quarterly, 9(1), 1-20, 2001.