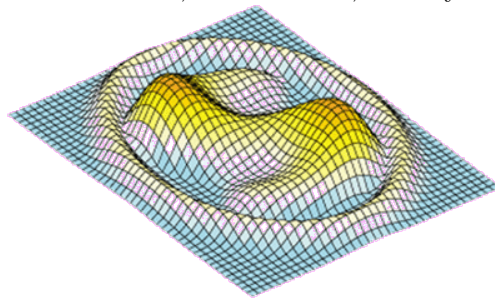


The Generalized National Airspace Trajectory-Prediction System

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.*

March 30, 2020

Contents

1	Introduction	1
2	Overall Architecture	3
2.1	The Server-Client Model	3
2.2	GNATS Interfaces	5
2.2.1	Equipment Interface	6
2.2.2	Environment Interface	9
2.2.3	Entity Interface	9
2.2.4	Simulation Interface	9
2.3	Safety Metrics	9
3	Datasets and Models	11
3.1	Aircraft Performance Model	11
3.2	National Airspace Data	11
3.3	Airport Surface Model	11
3.4	Terrain Data and Model	11
3.5	Wind Model and Data	11
3.6	Weather Data	11
3.7	Aircraft Cost and Load Data	11
4	Algorithms	13
4.1	Aircraft Propagation Algorithms	13
4.1.1	Nomenclature	13
4.1.2	Preliminaries	13
4.1.3	Propagation Algorithms for Each Phase	13
4.2	Taxiway Generation	30
4.3	Controller and Pilot Action	30
4.4	Conflict Detection and Resolution	31
4.4.1	Nomenclature	31
4.4.2	CDNR Algorithms	31
4.5	Weather Avoidance	35
4.6	Merging and Spacing	35
5	Setting Up and Running GNATS	39
5.1	Dependencies for Running GNATS	39
5.2	Flight Plan	39
5.3	The Server Client Structure	39
5.4	Multi-user Interface	39

6	Metrics	41
7	Example Use Cases	43

Chapter 1

Introduction

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

Chapter 2

Overall Architecture

The GNATS 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 GNATS 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 GNATS architecture and its critical components.

2.1 The Server-Client Model

The GNATS 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, JavaTM and Matlab[®].

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

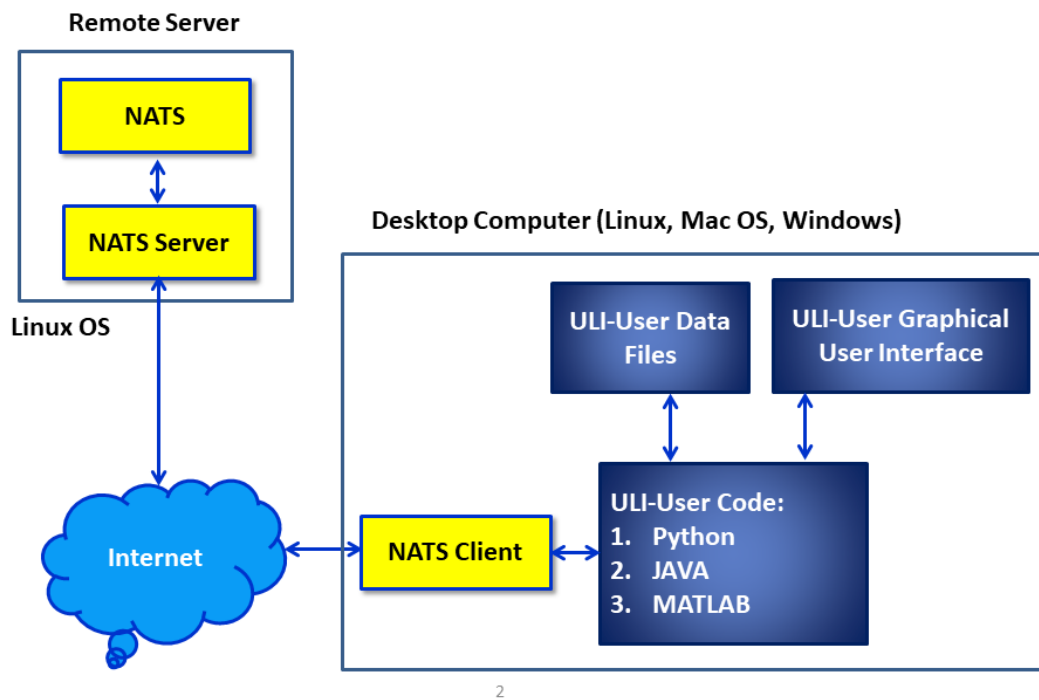


Figure 2.1: The Basic Structure of GNATS.

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

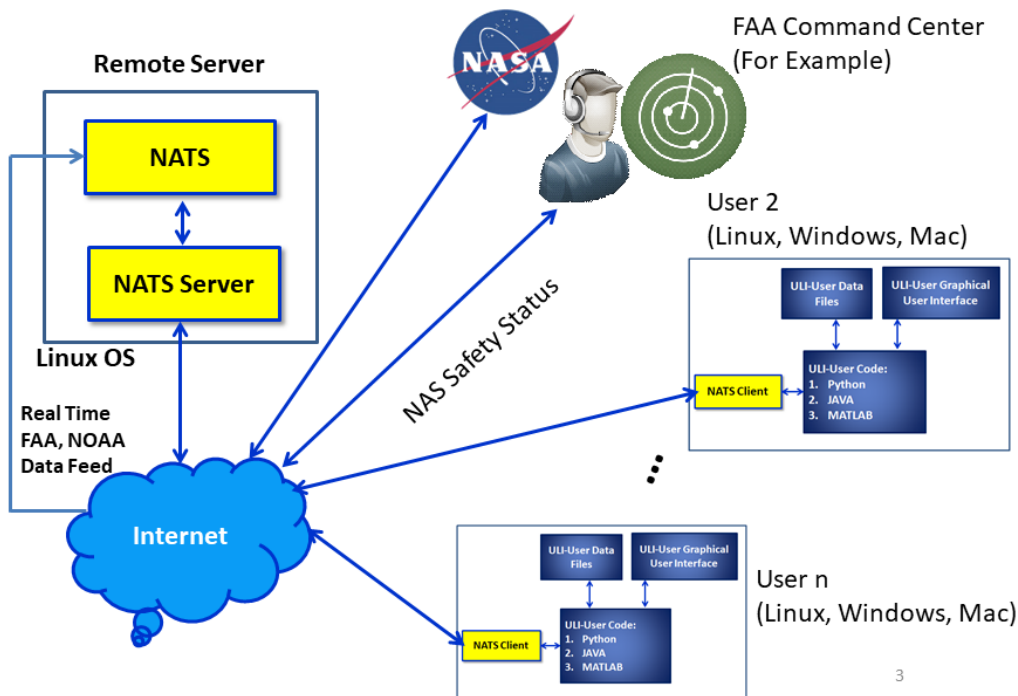


Figure 2.2: The Projected Capabilities of GNATS.

2.2 GNATS Interfaces

GNATS 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 oeperating staff.

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

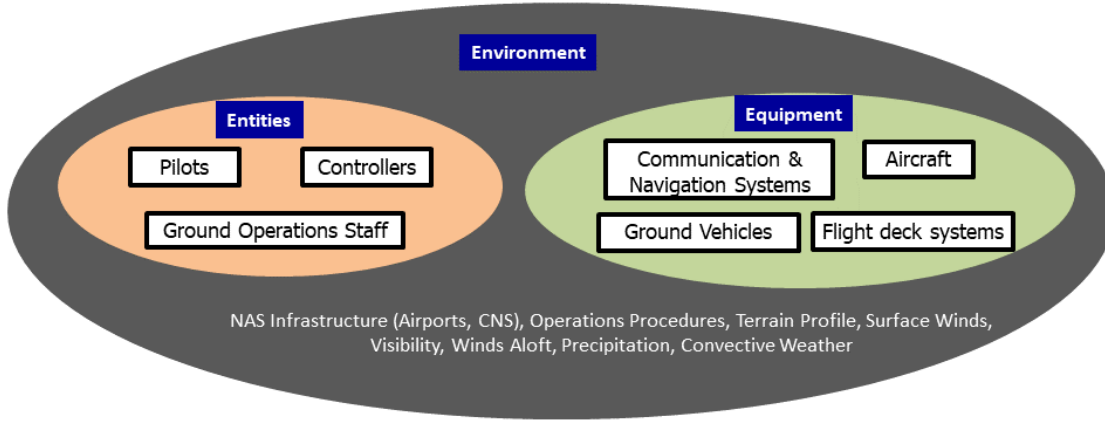


Figure 2.3: Different Components of the GNATS Software.

The GNATS 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:

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

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

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

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 GNATS 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:

$$\dot{\lambda} = \frac{V \cos \chi}{R_e + h} \quad (2.4)$$

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

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 GNATS 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 GNATS 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]:

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

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 GNATS 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. GNATS 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. GNATS 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 GNATS, 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. GNATS 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 GNATS 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 GNATS

3.5 Wind Model and Data

NOAA RAP and how it is used in GNATS

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 $h_2 \geq h_1$

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

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

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

4.1.3 Propagation Algorithms for Each Phase

Algorithm 1 Gate

- 1: Given $\chi = \chi_{gate}$
 - 2: **if** No pushback clearance **then**
 - 3: $v = 0, \dot{h} = 0$ and $\chi = \chi_{gate}$
 - 4: **end if**
-

Algorithm 2 Pushback

```

1: Given  $L_{pushback}, v_{tug}, \Delta t$  .
2:  $s = 0, v = 0, \chi = \chi_{gate}$ .
3:  $\dot{h} = 0, h = h_{apt}$  ▷  $h_{apt}$  is the altitude of airport.
4: while  $s < L_{pushback}$  do
5:   if no clearance received then
6:     continue
7:   else
8:      $v = v_{tug}$ 
9:   end if
10:  if  $L_{pushback} \leq s + v \Delta t$  then
11:     $s = L_{pushback}$ 
12:     $\chi = \chi_{ramp}$ 
13:    break
14:  else
15:     $s = s + v \Delta t$ 
16:     $\chi = \chi_{pushback}$  ▷ Use waypoint to waypoint tracking here
17:  end if
18: end while

```

Algorithm 3 Ramp

```

1: Given  $L_{ramp} = \sum_{i=1}^N L_{ramp_i}$  ▷ Sum of all ramp lengths for  $N$  ramp legs.
2: Given  $v_{ramp}, \Delta t$  .
3:  $s = 0, v = 0$ 
4:  $\dot{h} = 0, h = h_{apt}$  ▷  $h_{apt}$  is the altitude of airport.
5:  $\chi = \chi_{ramp_1}$ , ▷  $\chi_{ramp_1}$  is course of first rampway.
6:  $L_{tg} = L_{ramp}$  ▷  $L_{tg}$  is the length to go.
7:  $k = 1$  ▷ Ramp leg counter.
8: while  $L_{tg} > 0$  do
9:   Calculate  $L_{ramp_r}$  for every  $r = k, \dots, N$ . ▷ In case there is a perturbation during
   simulation this takes care of the everything.
10:   $L_{tg} = \sum_{j=k+1}^N L_{ramp_j} + d_{gc}(\lambda_c, \tau_c, h_c, \lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$  ▷  $(\lambda_c, \tau_c, h_c)$  is the current position and
    $(\lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$  is the next ramp waypoint.
11:  if no clearance received then
12:    continue
13:  else
14:     $v = v_{ramp}$ 
15:  end if
16:  if  $L_{tg} \leq v \Delta t$  then
17:     $s = L_{ramp}$ 
18:     $\chi = \chi_{ramp}$ , ▷  $\chi_{ramp}$  is the course at the end of ramp.
19:    break
20:  else
21:     $s = s + v \Delta t$ 
22:     $\chi = \chi_{ramp}(s)$ , ▷  $\chi_{ramp}(s)$  is the course at the current ramp position. Use waypoint to
   waypoint tracking
23:  end if
24:  if  $L_{tg} - v \Delta t \leq \sum_{j=k+1}^N L_{ramp_j}$  then
25:     $k = k + 1$ 
26:  end if
27: end while

```

Algorithm 4 Taxi

```

1: Given  $L_{taxi} = \sum_{i=1}^N L_{taxi_i}$  ▷ Sum of all taxi lengths for  $N$  taxi legs.
2: Given  $v_{taxi}, \Delta t$ .
3:  $s = 0, v = 0$ 
4:  $\dot{h} = 0, h = h_{apt}$  ▷  $h_{apt}$  is the altitude of airport
5:  $\chi = \chi_{taxi_1}$ , ▷  $\chi_{taxi_1}$  is course of first taxiway.
6:  $L_{tg} = L_{taxi}$  ▷  $L_{tg}$  is the length to go.
7:  $k = 1$  ▷ Taxi leg counter.
8: while  $L_{tg} > 0$  do
9:   Calculate  $L_{taxi_r}$  for every  $r = k, \dots, N$  ▷ In case there is a perturbation during simulation this takes care of the everything.
10:   $L_{tg} = \sum_{j=k+1}^N L_{taxi_j} + d_{gc}(\lambda_c, \tau_c, h_c, \lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$  ▷  $(\lambda_c, \tau_c, h_c)$  is the current position and  $(\lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$  is the next taxi waypoint.
11:   if no clearance received then
12:     continue
13:   else
14:      $v = v_{taxi}$ 
15:   end if
16:   if  $L_{tg} \leq v \Delta t$  then
17:      $s = L_{taxi}$ 
18:      $\chi = \chi_{rwy}$ , ▷  $\chi_{rwy}$  is runway course
19:     break
20:   else
21:      $s = s + v \Delta t$ 
22:      $\chi = \chi_{taxi}(s)$ , ▷  $\chi_{taxi}(s)$  is the course at the current taxi position. Use waypoint to waypoint tracking.
23:   end if
24:   if  $L_{tg} - v \Delta t \leq \sum_{j=k+1}^N L_{taxi_j}$  then
25:      $k = k + 1$ 
26:   end if
27: end while

```

Algorithm 5 Runway Threshold/ Hold Intersection

```

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

```

Algorithm 6 Takeoff and Maintain runway course leg

```

1: Given  $L_{takeoff}, v_{v_2}, \Delta t$  .
2: Given  $v_{wind} = [v_{dw}, v_{cw}]$   $\triangleright v_{dw}$  and  $v_{cw}$  are downwind and crosswind, respectively.
3: Given  $h = h_{wp1}$   $\triangleright h_{wp1}$  is the altitude of the first flight plan point after take off.
4:  $s = 0, v_t = 0$ 
5:  $\dot{h} = 0, h = h_{apt}$   $\triangleright h_{apt}$  is the altitude of airport
6:  $\chi = \chi_{rwy}$ 
7:  $a = \frac{(v_{v_2} - v_{dw})^2}{2 L_{takeoff}}$ 
8:  $L_{wp} = L_{takeoff}$ 
9: while  $h < h_{wp1}$  do
10:   if no takeoff clearance received then
11:      $v = 0, \dot{h} = 0, \chi = \chi_{rwy}$ 
12:     continue
13:   end if
14:   if  $\dot{h} = 0$  then
15:      $v = a \Delta t$   $\triangleright$  Use this rule before wheels off.
16:   else
17:      $\dot{h} = \dot{h}_{BADA}(h)$   $\triangleright$  Use BADA after takeoff. Linearly interpolate for  $\dot{h}$  for a given altitude
18:      $v_t = \{v_{CIFP}(h), v_{BADA}(h)\}$   $\triangleright$  If CIFP speed given use that or use BADA after takeoff.
        Linearly interpolate for  $v$  for a given altitude
19:      $\gamma = \sin^{-1} \left( \frac{\dot{h}}{v_t} \right)$   $\triangleright$  Flight path angle
20:      $v = v_t \cos \gamma$ 
21:   end if
22:   if  $L_{wp} \leq s + \frac{1}{2} a \Delta t^2$  and  $\dot{h} = 0$  then
23:      $\dot{h} = \dot{h}_{BADA}(h)$   $\triangleright$  Get initial  $\dot{h}$  from BADA
24:      $L_{wp} = L_{wp} + d_{gc}(\lambda_{v_2}, \tau_{v_2}, h_{v_2}, \lambda_{wp1}, \tau_{wp1}, h_{wp1})$   $\triangleright d_{gc}(\cdot)$  is the GC distance between  $v_2$ 
        point and the next waypoint in flight plan.
25:      $v_t = v_{v_2}$   $\triangleright$  Set speed as the  $v_2$  speed if  $L_{takeoff}$  reached between two time intervals.
26:      $\gamma = \sin^{-1} \left( \frac{\dot{h}}{v_t} \right)$   $\triangleright$  Flight path angle
27:      $v = v_t \cos \gamma$ 
28:      $s = L_{wp} + v_{v_2} \left( \Delta t - \sqrt{\frac{2(L_{wp} - s)}{a}} \right)$   $\triangleright$  Set  $s = L_{wp}$  the incremental distance flown if
         $L_{takeoff}$  reached between two time intervals.
29:      $h = h + \dot{h} \left( \Delta t - \sqrt{\frac{2(L_{wp} - s)}{a}} \right)$ 
30:     continue
31:   else if  $L_{wp} \leq s + v \Delta t$  and  $\dot{h} \neq 0$  then
32:      $h = h_{wp1}, s = L_{wp}$ 
33:     break
34:   else
35:     if  $\dot{h} = 0$  then
36:        $s = s + \frac{1}{2} a \Delta t^2$ 
37:     else
38:        $s = s + v \Delta t$ 
39:     end if
40:   end if
41:    $h = h + \dot{h} \Delta t$ 
42: end while

```

Algorithm 7 Climb out

```

1: Given  $h_{TRACON}, h_{maxBADA}$   $\triangleright$  Altitude of the TRACON for departing airport and the
   maximum altitude in BADA table less than  $h_{TRACON}$ , respectively
2: Given  $\Delta t, N$   $\triangleright$  Number of SID legs before  $h_{TRACON}$ .
3: Given  $L_{leg} = [L_{leg_1}, \dots, L_{leg_N}]$ ,  $\triangleright$  Length of each SID leg.
4: Given  $h_{wp_1}$   $\triangleright$   $h_{wp_1}$  is the altitude of the first flight plan point after take off.
5:  $h = h_{wp_1}, \dot{h} = \dot{h}_{BADA}(h_{wp_1})$ 
6:  $v_t = \{v_{CIFP}(h_{wp_1}), v_{BADA}(h_{wp_1})\}$   $\triangleright$  If CIFP speed given use that or use the BADA speed.
7:  $\chi = \chi_{rwy}$   $\triangleright$  Set  $\chi$  to the runway course
8:  $L_{tg} = \sum_{j=1}^N L_{leg_j}$   $\triangleright$   $L_{tg}$  is the length to go.
9:  $s = 0$   $\triangleright$  Distance along the SID
10:  $k = 0$   $\triangleright$  Leg counter
11: while  $h < h_{TRACON}$  do
12:    $v_t = \{v_{CIFP}(h), v_{BADA}(h)\}$ ,  $\triangleright$  Use Interpolation from BADA. If speed given in CIFP use
     that for interpolation limits.
13:   if  $h_{maxBADA} \leq h < h_{TRACON}$  then
14:     Use leveling off algorithm for  $\dot{h}$ 
15:   else
16:      $\dot{h} = \dot{h}_{BADA}(h)$ 
17:   end if
18:    $\gamma = \sin^{-1} \left( \frac{\dot{h}}{v_t} \right)$   $\triangleright$  Flight path angle
19:    $v = v_t \cos \gamma$ 
20:   if  $L_{tg} \leq v \Delta t$  then
21:      $s = \sum_{j=1}^N L_{leg_j}$ 
22:      $h = h_{TRACON}$ ,
23:      $\chi = \chi_{leg_N}$   $\triangleright$   $\chi_{leg_N}$  is the course of the last leg before  $h_{TRACON}$ 
24:     break
25:   end if
26:   Calculate  $L_{leg_r}$  for every  $r = k, \dots, N$   $\triangleright$  In case there is a perturbation during simulation
     this takes care of the everything.
27:    $L_{tg} = \sum_{j=k+1}^N L_{leg_j} + d_{gc}(\lambda_c, \tau_c, h_c, \lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$   $\triangleright$   $(\lambda_c, \tau_c, h_c)$  is the current position and
      $(\lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$  is the next flight plan waypoint.
28:    $h = h + \dot{h} \Delta t$ 
29:    $s = s + v \Delta t$ 
30:   if  $L_{tg} - v \Delta t \leq \sum_{i=k+1}^N L_{leg_i}$  then
31:      $k = k + 1$ 
32:   end if
33:    $\chi = \chi_{leg_k}$   $\triangleright$   $\chi_{leg_k}$  is the course of the  $k^{th}$  leg
34: end while

```

Algorithm 8 Hold in pattern after climb out

1: Given h_{TRACON} 2: Given L_H 3: Given χ_H 4: Given Δt 5: $v = v_{BADA}(h_{TRACON}), \dot{h} = 0$ 6: $s = 0, h = h_{TRACON}$ 7: while no clearance received do 8: $\chi = \chi_H$ 9: if $s + v\Delta t > L_H$ then 10: $s = s + v\Delta t - L_H$ 11: else 12: $s = s + v\Delta t$ 13: end if 14: end while	<div style="margin-left: 20px;">▷ Altitude of the TRACON for departing airport</div> <div style="margin-left: 20px;">▷ Length of the holding leg</div> <div style="margin-left: 20px;">▷ Course of the holding leg</div>
--	--

Algorithm 10 Top of climb

- 1: Given h_{cruise}, v_{cruise} \triangleright Cruise altitude and speed, respectively
 - 2: Given $(\lambda_{TOC}, \tau_{TOC})$ \triangleright Latitude and longitude of TOC
 - 3: **if** $h = h_{cruise}$ **then**
 - 4: $\dot{h} = 0$
 - 5: $v = v_{cruise}$
 - 6: $\chi = \chi_{gc}(\lambda_{TOC}, \tau_{TOC}, h_{cruise}, \lambda_{wp_1}, \tau_{wp_1}, h_{cruise})$ $\triangleright (\lambda_{wp_1}, \tau_{wp_1})$ are the latitude and longitude of the first cruise waypoint
 - 7: **end if**
-

Algorithm 11 Cruise

```

1: Given  $h_{cruise}, v_{cruise}$  ▷ Cruise altitude and speed, respectively
2: Given  $N$  ▷ Number of cruise legs
3: Given  $L_{leg} = [L_{leg_1}, \dots, L_{leg_N}]$ , ▷ Length of each cruise leg.
4: Given  $\Delta t$ 
5:  $h = h_{cruise}, v = v_{cruise}, \chi = \chi_{leg_1}, \dot{h} = 0$  ▷ Initial states
6:  $s = 0$ , ▷ Distance and leg counter
7: Set  $L = \sum_{k=1}^{N-1} L_{leg_k}$ 
8: Set  $L_{tg} = L$  ▷  $L_{tg}$  is the length to go.
9:  $k = 0$  ▷ Leg counter
10: while  $L_{tg} > 0$  do
11:   if  $L_{tg} \leq v \Delta t$  then
12:     if no clearance received then ▷ Maintain course and continue moving using the same
       speed and altitude
13:       Execute Algorithm 12
14:       continue
15:     else
16:        $s = L$ 
17:        $\chi = \chi_{leg_{N-1}}$ 
18:        $h = h_{cruise}$ 
19:       break
20:     end if
21:   end if
22:   Calculate  $L_{leg_r}$  for every  $r = k, \dots, N$  ▷ In case there is a perturbation during simulation
       this takes care of the everything.
23:    $L_{tg} = \sum_{j=k+1}^N L_{leg_j} + d_{gc}(\lambda_c, \tau_c, h_c, \lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$  ▷  $(\lambda_c, \tau_c, h_c)$  is the current position and
        $(\lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$  is the next flight plan waypoint.
24:    $s = s + v \Delta t$ 
25:   if  $L_{tg} - v \Delta t \leq \sum_{i=k+1}^N L_{leg_i}$  then
26:      $k = k + 1$ 
27:   end if
28:    $\chi = \chi_{leg_k}$  ▷  $\chi_{leg_k}$  is the course of the  $k^{th}$  leg
29: end while

```

Algorithm 12 Hold in pattern enroute

1: Given h_{cruise}, v_{cruise} 2: Given L_H 3: Given χ_H 4: Given Δt 5: $v = v_{cruise}, h = h_{cruise}, \dot{h} = 0$ 6: $s = 0, h = h_{cruise}$ 7: while no clearance received do 8: $\chi = \chi_H$ 9: if $s + v\Delta t > L_H$ then 10: $s = s + v\Delta t - L_H$ 11: else 12: $s = s + v\Delta t$ 13: end if 14: end while	<div style="margin-left: 20px;">▷ Cruise altitude and speed</div> <div style="margin-left: 20px;">▷ Length of the holding leg</div> <div style="margin-left: 20px;">▷ Course of the holding leg</div>
--	---

Algorithm 13 Top of descent

1: Given h_{cruise}, v_{cruise} 2: Given $(\lambda_{TOD}, \tau_{TOD})$ 3: Given χ_{leg_1} 4: Given Δt 5: if $h = h_{cruise}, \lambda = \lambda_{TOD}, \tau = \tau_{TOD}$ then 6: $\dot{h} = \dot{h}_{BADA}(h_{cruise})$ 7: $\chi = \chi_{leg_1}$ 8: $v_t = v_{BADA}(h_{cruise})$ 9: $\gamma = \sin^{-1} \left(\frac{\dot{h}}{v_t} \right)$ 10: $v = v_t \cos \gamma$ 11: end if	<div style="margin-left: 20px;">▷ Cruise altitude and speed, respectively</div> <div style="margin-left: 20px;">▷ Latitude and longitude of TOD</div> <div style="margin-left: 20px;">▷ χ_{leg_1} is the course along first STAR leg.</div> <div style="margin-left: 20px;">▷ Flight path angle</div>
--	---

Algorithm 14 Initial Descent

```

1: Given  $h_{TRACON}, h_{cruise}$   $\triangleright$  Altitude of the TRACON for arriving airport and cruise altitude,
   respectively
2: Given  $h_{minBADA} \triangleright h_{minBADA}$  is the minimum altitude in BADA table greater than  $h_{TRACON}$ 
3: Given  $h_{CA}$   $\triangleright$  Minimum altitude of the Class A airspace  $h_{CA} > h_{TRACON}$ 
4: Given  $\Delta t, N$   $\triangleright$  Number of STAR legs before  $h_{TRACON}$ 
5: Given  $L_{leg} = [L_{leg_1}, \dots, L_{leg_N}]$ ,  $\triangleright$  Length of each STAR Leg.
6:  $h = h_{cruise}, v_t = v_{cruise}, \chi = \chi_{leg_1}, \dot{h} = 0$   $\triangleright$  Initial states before descent
7:  $L_{tg} = \sum_{j=1}^N L_{leg_j}$   $\triangleright L_{tg}$  is the length to go.
8:  $s = 0$   $\triangleright$  Distance along the STAR
9:  $k = 0$   $\triangleright$  Leg counter
10: while  $L_{tg} > 0$  and  $h > h_{TRACON}$  do
11:    $v_t = \{v_{CIFP}(h), v_{BADA}(h)\}$ ,  $\triangleright$  Use Interpolation from BADA. If speed limit given in CIFP
   use that.
12:   if  $h_{TRACON} < h \leq h_{CA}$  then
13:     if no clearance received then
14:       Execute Algorithm 18.
15:     end if
16:   end if
17:   if  $h_{TRACON} < h \leq h_{minBADA}$  then
18:     Use leveling off algorithm for  $\dot{h}$ 
19:   else
20:      $\dot{h} = \dot{h}_{BADA}(h)$   $\triangleright$  Use Interpolation from BADA.
21:   end if
22:    $\gamma = \sin^{-1} \left( \frac{\dot{h}}{v_t} \right)$   $\triangleright$  Flight path angle
23:    $v = v_t \cos \gamma$ 
24:   if  $L_{tg} \leq v \Delta t$  then
25:      $s = \sum_{j=1}^N L_{leg_j}$ 
26:      $h = h_{TRACON}$ 
27:      $\chi = \chi_{leg_N}$ 
28:     break
29:   end if
30:   Calculate  $L_{leg_r}$  for every  $r = k, \dots, N$   $\triangleright$  In case there is a perturbation during simulation
   this takes care of the everything.
31:    $L_{tg} = \sum_{j=k+1}^N L_{leg_j} + d_{gc}(\lambda_c, \tau_c, h_c, \lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$   $\triangleright (\lambda_c, \tau_c, h_c)$  is the current position and
    $(\lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$  is the next flight plan waypoint.
32:    $d_{leg} = d_{gc}(\lambda_c, \tau_c, h_c, \lambda_{fp_k}, \tau_{fp_k}, h_{fp_k}), \theta = \frac{d_{leg}}{R_e + h}$   $\triangleright R_e$  is the radius of the earth.
33:    $\dot{s} = \theta v_t \sin \gamma + v_t \cos \gamma$ 
34:    $s = s + \dot{s} \Delta t$ 
35:    $h = h + \dot{h} \Delta t$ 
36:   if  $L_{tg} - v \Delta t \leq \sum_{i=k+1}^N L_{leg_i}$  then
37:      $k = k + 1$ 
38:   end if
39:    $\chi = \chi_{leg_k}$   $\triangleright \chi_{leg_k}$  is the course of the  $k^{th}$  leg
40: end while

```

Algorithm 15 Final Descent

```

1: Given  $h_{TRACON}, h_{IAF}$   $\triangleright$  Altitude of the TRACON for arriving airport and altitude of initial
   approach fix, respectively
2: Given  $\Delta t, N$   $\triangleright$  Number of STAR legs before  $h_{IAF}$  and after  $h_{TRACON}$ 
3: Given  $L_{leg} = [L_{leg_1}, \dots, L_{leg_N}]$ ,  $\triangleright$  Length of each STAR Leg.
4:  $h = h_{TRACON}$ ,  $\triangleright$  Initial altitude before final descent
5:  $v_t = \{v_{CIFP}(h_{TRACON}), v_{BADA}(h_{TRACON})\}$ ,  $\triangleright$  If CIPF speed given use that or use the
   BADA speed.
6:  $\dot{h} = \dot{h}_{BADA}(h)$ ,  $\gamma = \sin^{-1} \frac{\dot{h}}{v_t}$ ,  $v = v_t \cos \gamma$ 
7:  $L_{tg} = \sum_{j=1}^N L_{leg_j}$   $\triangleright L_{tg}$  is the length to go
8:  $s = 0$   $\triangleright$  Distance along the STAR
9:  $k = 0$   $\triangleright$  Leg counter
10: while  $L_{tg} > 0$  and  $h > h_{IAF}$  do
11:    $v_t = \{v_{CIFP}(h), v_{BADA}(h)\}$ ,  $\triangleright$  Use Interpolation from BADA. If speed limit given in CIPF
   use that.
12:    $\gamma = \gamma_{leg_k}$   $\triangleright$  Calculate flight path angle for  $k^{th}$  leg (See in eqn. 4.1)
13:    $\dot{h} = v_t \sin(\gamma)$   $\triangleright$  Calculate  $\dot{h}$  from FPA.
14:    $v = v_t \cos \gamma$ 
15:   if  $L_{tg} \leq v \Delta t$  then
16:      $s = \sum_{j=1}^N L_{leg_j}$ 
17:      $h = h_{IAF}$ 
18:      $\chi = \chi_{leg_N}$ 
19:     break
20:   end if
21:   Calculate  $L_{leg_r}$  for every  $r = k, \dots, N$   $\triangleright$  In case there is a perturbation during simulation
   this takes care of the everything.
22:    $L_{tg} = \sum_{j=k+1}^N L_{leg_j} + d_{gc}(\lambda_c, \tau_c, h_c, \lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$   $\triangleright (\lambda_c, \tau_c, h_c)$  is the current position and
    $(\lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$  is the next flight plan waypoint.
23:    $d_{leg} = d_{gc}(\lambda_c, \tau_c, h_c, \lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$ ,  $\theta = \frac{d_{leg}}{R_e + h}$   $\triangleright R_e$  is the radius of the earth.
24:    $\dot{s} = \theta v_t \sin \gamma + v_t \cos \gamma$ 
25:    $s = s + \dot{s} \Delta t$ 
26:    $h = h + \dot{h} \Delta t$ 
27:   if  $L_{tg} - v \Delta t \leq \sum_{i=k+1}^N L_{leg_i}$  then
28:      $k = k + 1$ 
29:   end if
30:    $\chi = \chi_{leg_k}$   $\triangleright \chi_{leg_k}$  is the course of the  $k^{th}$  leg
31: end while

```

Algorithm 16 Approach

-
- 1: Given h_{IAF} , h_{FAF} , h_{rtp} \triangleright Altitude of the of initial approach fix, final approach fix and runway touchdown point respectively
 - 2: Given v_{TD} \triangleright Touchdown speed of the aircraft.
 - 3: Given Δt , N \triangleright Final approach speed and altitude restriction points
 - 4: Given $L_{leg} = [L_{leg_1}, \dots, L_{leg_N}]$, \triangleright Length of each approach legs.
 - 5: $h = h_{IAF}$ \triangleright Initial altitude before approach
 - 6: $v_t = \{v_{CIFP}(h_{IAF}), v_{BADA}(h_{IAF})\}$, \triangleright If CIFP speed given use that or use the BADA speed.
 - 7: $\dot{h} = \dot{h}_\gamma(h)$, $v = v_t \cos \gamma$ $\triangleright \dot{h}_\gamma(h)$ is the \dot{h} at the flight path angle from previous phase. Similar arguments hold for v .
 - 8: $L_{tg} = \sum_{j=1}^N L_{leg_j}$ $\triangleright L_{tg}$ is the length to go
 - 9: $s = 0$ \triangleright Distance along the approach segment.
 - 10: $k = 0$ \triangleright Leg counter
 - 11: **while** $L_{tg} > 0$ **and** $h > h_{rwy}$ **do**
 - 12: $v_t = \{v_{CIFP}(h), v_{BADA}(h)\}$, \triangleright Use Interpolation from BADA. If speed limit given in CIFP use that.
 - 13: $\gamma = \gamma_{leg_k}$ \triangleright Calculate flight path angle for k^{th} leg (See in eqn. 4.1)
 - 14: $\dot{h} = v_t \sin(\gamma)$ \triangleright Calculate \dot{h} from FPA.
 - 15: $v = v_t \cos \gamma$
 - 16: **if** missed approach **then**
 - 17: Go to Algorithm 17 and then follow Algorithm 18
 - 18: **end if**
 - 19: **if** $h_{rtp} < h \leq h_{FAF}$ **then**
 - 20: Use leveling off algorithm for \dot{h} .
 - 21: **end if**
 - 22: **if** $L_{tg} \leq v \Delta t$ **then**
 - 23: $h = h_{rtp}$, $\dot{h} = 0$
 - 24: $s = \sum_{j=1}^N L_{leg_j}$
 - 25: $\chi = \chi_{rwy}$
 - 26: $v = v_{TD}$
 - 27: **break**
 - 28: **end if**
 - 29: Calculate L_{leg_r} for every $r = k, \dots, N$ \triangleright In case there is a perturbation during simulation this takes care of the everything.
 - 30: $L_{tg} = \sum_{j=k+1}^N L_{leg_j} + d_{gc}(\lambda_c, \tau_c, h_c, \lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$ $\triangleright (\lambda_c, \tau_c, h_c)$ is the current position and $(\lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$ is the next flight plan waypoint
 - 31: $d_{leg} = d_{gc}(\lambda_c, \tau_c, h_c, \lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$, $\theta = \frac{d_{leg}}{R_e + h}$ $\triangleright R_e$ is the radius of the earth.
 - 32: $\dot{s} = \theta v_t \sin \gamma + v_t \cos \gamma$
 - 33: $s = s + \dot{s} \Delta t$
 - 34: **if** $L_{tg} - v \Delta t \leq \sum_{i=k+1}^N L_{leg_i}$ **then**
 - 35: $h = h + \dot{h} \Delta t$
 - 36: $k = k + 1$
 - 37: **end if**
 - 38: $\chi = \chi_{leg_k}$ $\triangleright \chi_{leg_k}$ is the course of the k^{th} leg
 - 39: **end while**
-

Algorithm 17 Go around for missed approach

```

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

```

Algorithm 18 Hold in descent

```

1: Given  $h_H, v_H$                                 ▷ Altitude and speed of the holding pattern
2: Given  $\chi_H$                                 ▷ Course of the holding pattern leg
3: Given  $\Delta t$ 
4:  $\dot{h} = 0$ 
5:  $h = h_H, s = 0$ 
6: while no clearance received do
7:    $v = \{v_{CIFP}(h), v_H\}$ , ▷ Use Interpolation from BADA. If speed limit given in CIFP use
      that.
8:    $\chi = \chi_H$ 
9:   if  $s + v \Delta t > L_H$  then
10:      $s = s + v \Delta t - L_H$ 
11:   else
12:      $s = s + v \Delta t$ 
13:   end if
14: end while

```

Algorithm 19 Touchdown

```

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

```

Algorithm 20 Landing to stop

```

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

```

Algorithm 21 Runway exit

```

1: Given  $\chi_{taxi}$                                 ▷ Given course of first taxiway
2: Given  $v_{taxi}$                                 ▷ Speed of taxi.
3: Given  $L_{re}$                                 ▷ Length of the arc needed for transition from runway to taxiway.
4:  $\dot{h} = 0, \chi = \chi_{rwy}, v = v_{taxi}$                 ▷ Initial conditions
5:  $s = 0, h = h_{apt}$                             ▷  $h_{apt}$  is the altitude of airport
6: while  $\chi \neq \chi_{taxi}$  and  $s < L_{re}$  do
7:   if  $s + v \Delta t > L_{re}$  then
8:      $s = L_{re}$ 
9:      $\chi = \chi_{taxi}$ 
10:    break
11:   end if
12:    $\chi = \chi_{rwy} + \frac{\chi_{taxi} - \chi_{rwy}}{L_{re}} s$ 
13:    $s = s + v \Delta t$ 
14: end while

```

Algorithm 22 Taxi

```

1: Given  $L_{taxi} = \sum_{i=1}^N L_{taxi_i}$  ▷ Sum of all taxi lengths for  $N$  taxi legs.
2: Given  $v_{taxi}, \Delta t$  .
3:  $s = 0, v = 0$ 
4:  $\dot{h} = 0, h = h_{apt}$  ▷  $h_{apt}$  is the altitude of airport
5:  $\chi = \chi_{taxi_1},$  ▷  $\chi_{taxi_1}$  is course of first taxiway.
6:  $L_{tg} = L_{taxi}$  ▷  $L_{tg}$  is the length to go.
7:  $k = 1$  ▷ Taxi leg counter.
8: while  $L_{tg} > 0$  do
9:   Calculate  $L_{taxi_r}$  for every  $r = k, \dots, N$  ▷ In case there is a perturbation during simulation this takes care of the everything.
10:   $L_{tg} = \sum_{j=k+1}^N L_{taxi_j} + d_{gc}(\lambda_c, \tau_c, h_c, \lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$  ▷  $(\lambda_c, \tau_c, h_c)$  is the current position and  $(\lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$  is the next taxi waypoint.
11:   if no clearance received then
12:     continue
13:   else
14:      $v = v_{taxi}$ 
15:   end if
16:   if  $L_{tg} \leq v \Delta t$  then
17:      $s = L_{taxi}$ 
18:      $\chi = \chi_{ramp_1},$  ▷  $\chi_{rwy}$  is runway course
19:     break
20:   else
21:      $s = s + v \Delta t$ 
22:      $\chi = \chi_{taxi}(s),$  ▷  $\chi_{taxi}(s)$  is the course at the current taxi position. Use waypoint to waypoint tracking.
23:   end if
24:   if  $L_{tg} - v \Delta t \leq \sum_{j=k+1}^N L_{taxi_j}$  then
25:      $k = k + 1$ 
26:   end if
27: end while

```

Algorithm 23 Runway Threshold/ Hold Intersection

```

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

```

Algorithm 24 Ramp

```

1: Given  $v_{ramp}, \Delta t$  .
2:  $s = 0, v = 0$ 
3:  $\dot{h} = 0, h = h_{apt}$   $\triangleright h_{apt}$  is the altitude of airport.
4:  $\chi = \chi_{ramp_1}$ ,  $\triangleright \chi_{ramp_1}$  is course of first rampway.
5:  $L_{tg} = L_{ramp}$   $\triangleright L_{tg}$  is the length to go.
6:  $k = 1$   $\triangleright$  Ramp leg counter.
7: while  $L_{tg} > 0$  do
8:   Calculate  $L_{ramp_r}$  for every  $r = k, \dots, N$ .  $\triangleright$  In case there is a perturbation during
   simulation this takes care of the everything.
9:    $L_{tg} = \sum_{j=k+1}^N L_{ramp_j} + d_{gc}(\lambda_c, \tau_c, h_c, \lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$   $\triangleright (\lambda_c, \tau_c, h_c)$  is the current position and
    $(\lambda_{fp_k}, \tau_{fp_k}, h_{fp_k})$  is the next ramp waypoint.
10:  if no clearance received then
11:    continue
12:  else
13:     $v = v_{ramp}$ 
14:  end if
15:  if  $L_{tg} \leq v \Delta t$  then
16:     $s = L_{ramp}$ 
17:     $\chi = \chi_{gate}$ ,  $\triangleright \chi_{gate}$  is the course at of the gate.
18:  else
19:     $s = s + v \Delta t$ 
20:     $\chi = \chi_{ramp}(s)$ ,  $\triangleright \chi_{ramp}(s)$  is the course at the current ramp position. Use waypoint to
    waypoint tracking
21:  end if
22:  if  $L_{tg} - v \Delta t \leq \sum_{j=k+1}^N L_{ramp_j}$  then
23:     $k = k + 1$ 
24:  end if
25: end while

```

Algorithm 25 Gate

```

1: Given  $\chi = \chi_{gate}$ 
2:  $v = 0, \dot{h} = 0$  and  $\chi = \chi_{gate}$ 
3:  $h = h_{apt}$   $\triangleright h_{apt}$  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

```

1: procedure GEODETICTOECEFCONVERSION( $\lambda, \tau, h, V, \gamma, \chi, R_e$ )
2:    $\dot{x}_{ecef} = V \sin \lambda \cos \tau \cos \gamma \cos \chi - V \sin \tau \cos \gamma \sin \chi + V \cos \lambda \cos \tau \sin \gamma$ 
3:    $\dot{y}_{ecef} = V \sin \lambda \sin \tau \cos \gamma \cos \chi + V \cos \tau \cos \gamma \sin \chi + V \cos \lambda \sin \tau \sin \gamma$ 
4:    $\dot{z}_{ecef} = -V \cos \lambda \cos \gamma \cos \chi + V \sin \lambda \sin \gamma$ 
5:    $x_{ecef} = (R_e + h) \cos \lambda \cos \tau$ 
6:    $y_{ecef} = (R_e + h) \cos \lambda \sin \tau$ 
7:    $z_{ecef} = (R_e + h) \sin \lambda$ 
8:   return  $x_{ecef}, y_{ecef}, z_{ecef}, \dot{x}_{ecef}, \dot{y}_{ecef}, \dot{z}_{ecef}$ 
9: end procedure

```

Algorithm 27 Get A and B

```

1: procedure GETAANDBCOEFFICIENTS( $x, y, z, \dot{x}, \dot{y}, \dot{z}$ )
2:    $D = x\dot{y} - y\dot{x}$ 
3:   if  $D = 0$  then
4:     Denominator is 0. Exiting
5:     return 0, 0
6:   end if
7:    $A = \frac{\dot{y}z - y\dot{z}}{D}$ 
8:    $B = \frac{\dot{z}x - z\dot{x}}{D}$ 
9:   return  $A, B$ 
10: end procedure

```

Algorithm 28 Get Conflict Point

```

1: procedure GETCONFLICTPOINT( $A_1, B_1, A_2, B_2, h, R_e$ )
2:    $z_c = (R_e + h) \sqrt{\frac{1}{1 + \left(\frac{A_1 - A_2}{A_1 B_2 - A_2 B_1}\right)^2 + \left(\frac{B_2 - B_1}{A_1 B_2 - A_2 B_1}\right)^2}}$ 
3:    $y_c = \frac{A_1 - A_2}{A_1 B_2 - A_2 B_1} z_c$ 
4:    $x_c = \frac{B_2 - B_1}{A_1 B_2 - A_2 B_1} z_c$ 
5:   return  $x_c, y_c, z_c$ 
6: end procedure

```

Algorithm 29 Get Time To Intersection

```

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

```

Algorithm 30 Detect Conflict to Go

```

1: procedure DETECTCONFLICTTOGO( $\lambda_1, \tau_1, h_1, V_1, \gamma_1, \chi_1, \lambda_2, \tau_2, h_2, V_2, \gamma_2, \chi_2, R_e$ )
2:    $h = \frac{h_1 + h_2}{2}$  ▷ Placeholder.
3:    $x_1, y_1, z_1, \dot{x}_1, \dot{y}_1, \dot{z}_1 = \text{GeodeticToECEFConversion}(\lambda_1, \tau_1, h_1, V_1, \gamma_1, \chi_1, R_e)$ 
4:    $x_2, y_2, z_2, \dot{x}_2, \dot{y}_2, \dot{z}_2 = \text{GeodeticToECEFConversion}(\lambda_2, \tau_2, h_2, V_2, \gamma_2, \chi_2, R_e)$ 
5:    $A_1, B_1 = \text{GetAandBCoefficients}(x_1, y_1, z_1, \dot{x}_1, \dot{y}_1, \dot{z}_1)$ 
6:    $A_2, B_2 = \text{GetAandBCoefficients}(x_2, y_2, z_2, \dot{x}_2, \dot{y}_2, \dot{z}_2)$ 
7:    $x_c, y_c, z_c = \text{GetConflictPoint}(A_1, B_1, A_2, B_2, h, R_e)$ 
8:    $t_{c1} = \text{GetTimeToIntersectionPoint}(x_1, y_1, z_1, x_c, y_c, z_c, V_1, h, R_e)$ 
9:    $t_{c2} = \text{GetTimeToIntersectionPoint}(x_2, y_2, z_2, x_c, y_c, z_c, V_2, h, R_e)$ 
10:  return True,  $t_{c2}, t_{c1}$ 
11: end procedure

```

Algorithm 31 Conflict Detection

```

1: procedure CONFLICTDETECTION( $\lambda_1, \tau_1, h_1, V_1, \gamma_1, \chi_1, \lambda_2, \tau_2, h_2, V_2, \gamma_2, \chi_2, R_e, d_{detect}$ )
2:   if  $h_1 < 29000ft$  and  $h_2 < 29000ft$  then
3:     if  $\|h_1 - h_2\| > 1000$  then
4:       return False,-1,-1
5:     end if
6:   else if  $h_1 < 29000ft$  and  $h_2 \geq 29000ft$  or  $h_1 \geq 29000ft$  and  $h_2 < 29000ft$  then
7:     if  $\|h_1 - h_2\| > 1000$  then
8:       return False,-1,-1
9:     end if
10:  else
11:    if  $\|h_1 - h_2\| > 2000$  then
12:      return False,-1,-1
13:    end if
14:  end if
15:   $h = \frac{h_1 + h_2}{2}$  ▷ Placeholder.
16:   $d_{gc} = d_{gc}(\lambda_1, \tau_1, h, \lambda_2, \tau_2, h)$  ▷ Great circle distance
17:  if  $d_{gc} > d_{initiate}$  then
18:    return False,-1,-1
19:  end if
20:  return DetectConflictToGo( $\lambda_1, \tau_1, h_1, V_1, \gamma_1, \chi_1, \lambda_2, \tau_2, h_2, V_2, \gamma_2, \chi_2, R_e$ )
21: end procedure

```

Algorithm 32 Number of time steps to hold

```

1: procedure FINDNUMBEROFTIMESTEPSTO HOLD( $d_l, d_h, V, \Delta t$ )
2:    $\Delta D = d_h - d_l$ 
3:    $t = \Delta D / V$ 
4:    $n_s = t / \Delta t$ 
5:    $n_s = \text{floor}(n_s) + 1$ 
6:   return  $n_s$ 
7: end procedure

```

Algorithm 33 Conflict Resolution

```

1: procedure CONFLICTRESOLUTION( $V_1, V_2, R_e, t_{c1}, t_{c2}, d_{separation}$ )
2:   if  $t_{c1} > t_{c2}$  then
3:     if  $(t_{c1} - t_{c2})V_1 < d_{separation}$  then
4:       return 1, FindNumberOfTimeStepsToHold(  $(t_{c1} - t_{c2})V_1, d_{separation}, V_1, \Delta t$ )
5:     end if
6:   else if  $t_{c1} < t_{c2}$  then
7:     if  $(t_{c2} - t_{c1})V_2 < d_{separation}$  then
8:       return 2, FindNumberOfTimeStepsToHold(  $(t_{c2} - t_{c1})V_2, d_{separation}, V_2, \Delta t$ )
9:     end if
10:  else
11:     $t_1 = \frac{t_{c1}V_1 - d_{separation}}{V_1}$ 
12:     $t_2 = \frac{t_{c2}V_2 - d_{separation}}{V_2}$ 
13:    if  $t_1 > t_2$  then
14:      return 1,  $\text{floor}(t_2/\Delta t) + 1$ 
15:    else if  $t_1 < t_2$  then
16:      return 2,  $\text{floor}(t_1/\Delta t) + 1$ 
17:    else
18:      return 1, 1.
19:    end if
20:  end if
21: end procedure

```

Algorithm 34 CDNR

```

1: procedure CONFLICTDETECTIONANDRESOLUTION( $\lambda_1, \tau_1, h_1, V_1, \gamma_1, \chi_1, \lambda_2, \tau_2, h_2, V_2, \gamma_2, \chi_2, R_e, d_{initiate}, d_{separation}$ )
2:    $\text{IsConflict}, t_{c1}, t_{c2} = \text{ConflictDetection}(\lambda_1, \tau_1, h_1, V_1, \gamma_1, \chi_1, \lambda_2, \tau_2, h_2, V_2, \gamma_2, \chi_2, R_e, d_{initiate})$ 
3:   if IsConflict then
4:      $d_{ac}, d_{step} = \text{ConflictResolution}(V_1, V_2, R_e, t_{c1}, t_{c2}, d_{separation})$ 
5:     return  $d_{ac}, d_{step}$ 
6:   else
7:     Do nothing no conflict
8:     return -1, -1
9:   end if
10: end procedure

```

Algorithm 35 Propagation

```

1: Given  $N$ - number of aircraft,  $t_0$ ,  $t_f$  and  $\Delta t$  - Propagation time parameters.
2: procedure PROPAGATION( $N, t_0, t_f, \Delta t$ )
3:    $t = t_0$ 
4:   while  $t < t_f$  do
5:     for  $n = 1 \mapsto N$  do
6:       Propagate aircraft  $n$ 
7:     end for
8:     for  $n = 2 \mapsto N$  do
9:       for  $m = 1 \mapsto n - 1$  do
10:         $d_{ac}, d_{step} = \text{ConflictDetectionAndResolution}(\lambda_n, \tau_n, h_n, V_n, \gamma_n, \chi_n, \lambda_m, \tau_m, h_m,$ 
11:         $V_m, \gamma_m, \chi_m, R_e, d_{initiate}, d_{separation})$ 
12:        if  $d_{ac} == 1$  then
13:          Delay AC  $n$  for  $d_{step}$  time intervals.
14:          Skip CDNR for pair  $(m, n)$  for next  $d_{step}$  time intervals.
15:        else if  $d_{ac} == 2$  then
16:          Delay AC  $m$  for  $d_{step}$  time intervals.
17:          Skip CDNR for pair  $(m, n)$  for next  $d_{step}$  time intervals.
18:        else
19:          continue
20:        end if
21:      end for
22:    end for
23:  end while
24: 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 GNATS.

Algorithm 36 GetMeterFixPoints

```

1: procedure GETMETERFIXPOINTS(aplist[N],starlist[N, M])
2:   meterFixMap := ap  $\mapsto$  fixlist
3:   for n = 1  $\mapsto$  N do
4:     meterFixForAp = ▷ Vector of all meterfixes for the airport.
5:     for m = 1  $\mapsto$  M do
6:       star = starlist[n, m]
7:       wp = getString(star) ▷ Get all strings preceding the number in star
8:       if wp  $\in$  getWaypoints(star) then ▷ Find if the waypoint wp in list of waypoints in
          star.
9:         meterFixForAp = [meterFixForAp, wp] ▷ Append the wp to meter fix list.
10:      end if
11:    end for
12:    meterFixMap := aplist[n]  $\mapsto$  meterFixForAp ▷ Insert in the meterfix map.
13:  end for
14:  return meterFixMap.
15: end procedure

```

Algorithm 37 CreateMeterfixToAcAndDistMap

```

1: procedure CREATETERFIXTOACANDDISTMAP(ac,destAp,aar,acTargWp,meterFixMap,
    $\lambda_{targwp}$ , $\tau_{targwp}$ , $h_{targwp}$ , $\lambda_{curr}$ , $\tau_{curr}$ , $h_{curr}$ , meterfixtoAcAndDistMap)
2:    $\Delta t_{aar} = \frac{1}{aar}$ 
3:   if acTargWp in meterFixMap[destAp] then
4:     dtargwp = dgc( $\lambda_{targwp}$ , $\tau_{targwp}$ , $h_{targwp}$ , $\lambda_{curr}$ , $\tau_{curr}$ , $h_{curr}$ )
5:     if  $\Delta t_{aar} < d_{targwp}/V_{GS}$  then
6:       dist_ac_pair = (ac, dtargwp)
7:       meterfixtoAcAndDistMap[acTargWp].push_back(dist_ac_pair)
8:     end if
9:   end if
10: end procedure

```

Algorithm 38 GetACtoHold

```

1: procedure GETACTOHOLD(meterfixtoAcAndDistMap, acList)
2:   Given number of aircraft  $N$ .
3:    $acToHold = ac \mapsto bool$   $\triangleright map < string, bool >$ 
4:   for  $n = 1 \mapsto N$  do
5:      $acToHold[acList[n]] = False$   $\triangleright$  By default we will not hold any aircraft.
6:   end for
7:   for  $iter = meterfixtoAcAndDistMap.begin() \mapsto meterfixtoAcAndDistMap.end()$  do
8:      $meterfixPoint = iter \mapsto first$ 
9:      $vectorOfAcAndDist = iter \mapsto second$ 
10:     $M = vectorOfAcAndDist.size()$ 
11:     $MinDist = max\_numericlimit\_double$ 
12:     $ACNotToHold = NONE$ 
13:    for  $m = 1 \mapsto M$  do
14:       $ac\_dist\_pair = vectorOfAcAndDist[m]$ 
15:       $ac = ac\_dist\_pair.first$ 
16:       $dist = ac\_dist\_pair.second$ 
17:      if  $MinDist > dist$  then
18:         $MinDist = dist$ 
19:         $ACNotToHold = ac$ 
20:      end if
21:    end for
22:    if  $ACNotToHold \neq NONE$  then
23:      for  $m = 1 \mapsto M$  do
24:         $ac\_dist\_pair = vectorOfAcAndDist[m]$ 
25:         $ac = ac\_dist\_pair.first$ 
26:        if  $ACNotToHold \neq ac$  then
27:           $acToHold[ac] = True$ 
28:        end if
29:      end for
30:    end if
31:  end for
32:  return  $acToHold$ 
33: end procedure

```

Algorithm 39 Propagation

```

1: Given  $N$ - number of aircraft,  $t_0$ ,  $t_f$  and  $\Delta t$  - Propagation time parameters.
2: procedure PROPAGATION( $N, t_0, t_f, \Delta t$ )
3:    $t = t_0$ 
4:   while  $t < t_f$  do
5:      $meterfixtoAcAndDistMap := fixlist \mapsto vector < pair < ac, dist > >$   $\triangleright$  Map of all fixes
        to all aircrafts and their corresponding distances to the meter fix.
6:     for  $n = 1 \mapsto N$  do
7:        $CreateMeterfixToAcAndDistMap(aclist[n], destAp, aar, acTargWp, meterFixMap, \lambda_{targwp}, \tau_{targwp},$ 
         $meterfixtoAcAndDistMap)$   $\triangleright$  Create a map of distance to meterfix and corresponding ac
8:     end for
9:      $acToHold = GetACtoHold(meterfixtoAcAndDistMap, aclist)$   $\triangleright$ 
         $acToHold = ac \mapsto bool$  gives the list of acs to hold
10:    for  $n = 1 \mapsto N$  do
11:      if  $actoHold[aclist[n]] == \text{False}$  then
12:        Propagate aircraft  $n$ 
13:      end if
14:      Calculate states at  $t + \Delta t$ .
15:    end for
16:     $t = t + \Delta t$ 
17:  end while
18: end procedure

```

Algorithm 40 GNATS Simulation

```

1: procedure PREPROCESSING
2:    $meterFixMap = GetMeterFixPoints(aplist[N], starlist[N, M])$ 
3: end procedure
4: Propagation( $meterFixMap, \dots$ )

```

Chapter 5

Setting Up and Running GNATS

All the sections in this Chapter are to be completed.

5.1 Dependencies for Running GNATS

Prerequisites for running GNATS.

5.2 Flight Plan

How to create a flight plan for GNATS 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 GNATS

Chapter 7

Example Use Cases

Provide some example use cases of GNATS.

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.