JETSTREAM MODEL

ICCT JETSTREAM model v0.11 documentation

8/18/2025

THE INTERNATIONAL COUNCIL ON CLEAN TRANSPORTATION

Table of Contents

Acknowledgements

JETSTREAM was first developed in 2024 by Jonathan Benoit, Jayant Mukhopadhaya, Deniz Rhode, Uwe Tietge, and Daniel Sitompul.

Introduction

JETSTREAM, short for Jet and Turboprop Simulations for Trajectory-based Emissions and Meteorological Effects, is the ICCT's aviation emissions model. It uses aircraft trajectories, meteorological data, and aircraft and engine performance models to estimate global emissions resulting from aircraft activity. It also utilizes a contrail process model to estimate the climate impact of contrail cirrus produced during flight. The results are then compared to reported emissions to validate the accuracy of the modeling process.

This work serves as the model documentation for JETSTREAM. It begins by describing the datasets used, along with any cleaning steps taken to prepare the data for input into the models. It then describes the modeling steps taken to convert the data inputs into a comprehensive inventory of aviation's CO_2 and non-CO_2 emissions. The results of the inventory are then compared to publicly available emissions reports and other similar modeling efforts.

Overall Model Schematic

JETSTREAM operates as a wrapper around the pycontrails aircraft emissions and contrails modeling software. Pycontrails uses:

1. **aircraft trajectories**, defined as a time series of latitude, longitude, and altitude,
2. **aircraft type,** defined by the International Civil Aviation Organization (ICAO) type designator,
3. **engine type**, defined by an Engine unique identifier (UID) from the ICAO Engine Emissions Databank (EEDB),
4. **atmospheric conditions,** defined as temperature, humidity, pressure, wind speeds, cloud cover, incoming short-wave radiation, and outgoing long-wave radiation,

as input data for its aircraft performance, engine emissions, and contrail impact models. The results of the modeling are then validated against real-world data to test the quality of the modeling. The overall schematic is shown in Figure 1.

The aircraft performance model utilizes aircraft trajectories, aircraft type information, and atmospheric conditions to calculate the fuel consumed at each point along the flight trajectory. The fuel burn estimates feed into the engine emissions model, which uses the engine type, its emissions certification results from the ICAO EEDB, and atmospheric conditions, to estimate the emission quantities of CO_2, Nitrogen oxides (NO_x), non-volatile particulate matter (nvPM), sulfur oxides (SO_x), and water vapor. The emission quantities (specifically, nvPM and water vapor), along with engine, aircraft, and atmospheric data, are fed into the Contrail Cirrus Prediction (CoCiP) model, which simulates the entire lifetime of any contrail that may form and estimates the radiative forcing that it creates.

The emission estimates are validated against available real-world datasets, such as country-pair emissions reported to CORSIA and flight-specific fuel consumption data reported to Brazil's ANAC (Agência Nacional de Aviação Civil). We also match the flight inventory with OAG's annual flight scheduling data to obtain a more accurate estimate of emissions from commercial aviation.

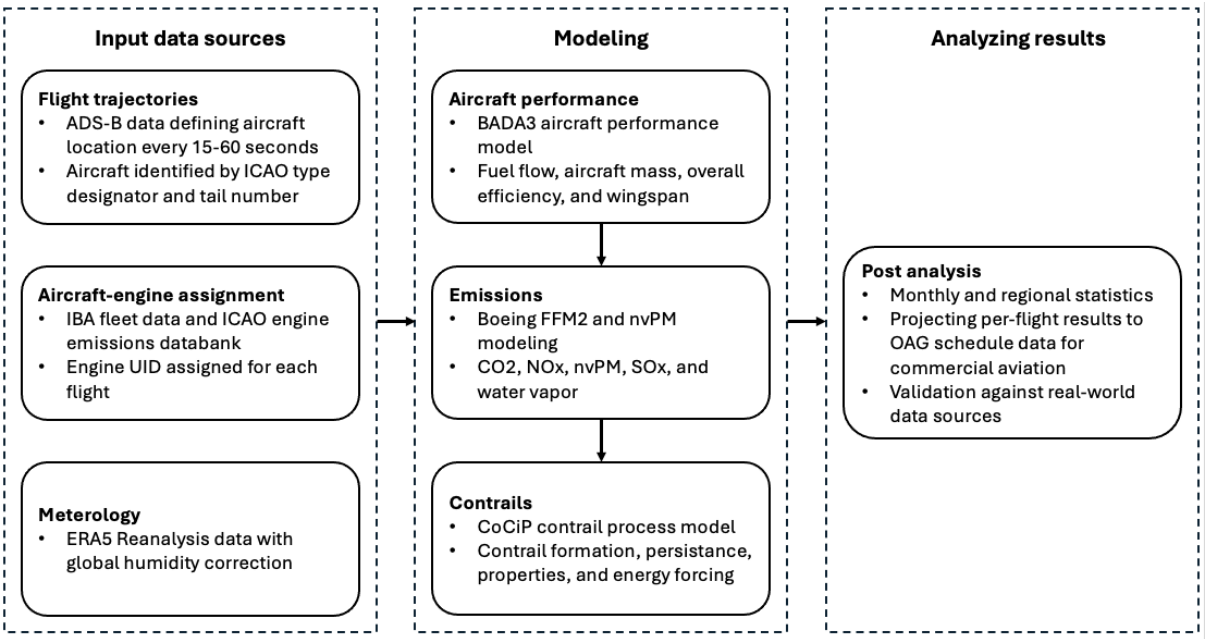


Figure 1 Schematic of the JETSTREAM modeling process

Datasets

Flight trajectory data

Description

Data provider: Spire

Most aircraft broadcast their position in real-time to ground and satellite receivers through a technology called Automatic Dependent Surveillance-Broadcast (ADS-B).

We use Spire's ADS-B database as our primary data source for flight trajectories. The dataset includes trajectory data for 62.0 million global flight movements. It consists of all flights that were equipped with ADS-B transmitters, captured by ground or satellite receivers. Each flight is identified by a unique $flight\_id$ and has a trajectory associated metadata. The description of the available fields is documented in the Spire API documentation.

The trajectory files are stored in Azure Blob storage rather than in a database to save on storage costs. The delivered data has a custom adaptive downsample based on the duration of the flight.

1. For flights less than 500 km in length, or with less than 45 minutes of airborne time, a sample rate of 15 seconds is used.
2. For flights less than 1500 km in length, or with less than 1.5 hours of airborne time, a sample rate of 30 seconds is used.
3. For all other flights, a sample rate of 1 min is used.

The trajectory data is not perfect. While ADS-B transmitters onboard aircraft are constantly broadcasting their location, it is not always picked up by a receiver. This creates data gaps in areas where there are insufficient ADS-B receivers, such as over the ocean or in uninhabited places.

The metadata provides information about the airline, including arrival and departure airports, as well as the aircraft type, registration, and role. The metadata is stored in a database. The fields and their fill rates are shown in Figure 2. $flight\_id$, $observation\_start$, $observation\_end$, $icao\_address$, and $duration$ are the only fields with a 100% fill rate. Aircraft types are identified using a combination of $icao\_address$, ac_type_icao, and ac_type_name (details in the **Error! Reference source not found.** section). The departure and arrival airports are determined based on a combination of $dep\_airport\_icao$, $arr\_airport\_icao$, $dep\_airport\_iata$, and $arr\_airport\_icao$ (details in the Airport data section). There are additional airport code variables with the $provider\_$ and $spire\_$ prefixes. These fields distinguish between the airport codes supplied by Spire's data provider and those detected by Spire themselves. We use the airport code variables without the prefix, as these are the most complete. The airline information is not used in the emissions modeling, but is used in combining the Spire trajectory data and the OAG flight schedule data. The airline is determined based on a combination of $airline\_iata$ and $airline\_name$.

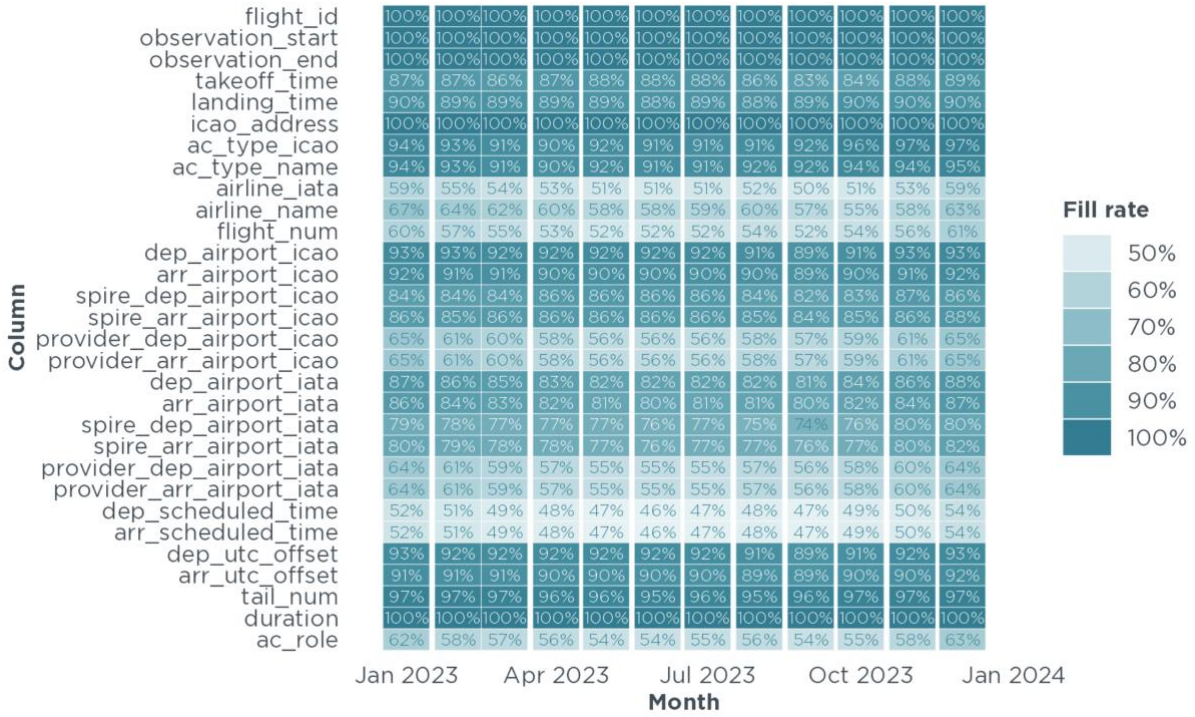


Figure 2 Data fill rate for the flight trajectory metadata

The metadata is not perfect, and conflicting information needs to be resolved. Additionally, mapping between ICAO codes and IATA codes for aircraft and airports is challenging as there can be a many-to-many mapping relationship between the IATA and ICAO codes. These discrepancies need to be resolved because the aircraft performance models utilize ICAO codes for aircraft types, whereas the schedule data from OAG uses IATA codes for the same types. Similarly, OAG uses IATA codes for airports. However, for modeling purposes, we use airport ICAO codes as there are far more airports with assigned ICAO codes than those with assigned IATA codes.

The following sections outline the data cleaning processes undertaken prior to running the simulations.

Preprocessing Trajectories

The trajectory preprocessing happens in two steps:

- Initial trajectory classification and interpolation/extrapolation.

- Secondary trajectory cleaning to correct unrealistic trajectories.

Initial trajectory classification and completion

The trajectories are classified as:

1. *FULL*: Starting and ending points are below 3000 ft, and the time between successive waypoints is always less than 15 minutes
2. *INTR*: Some interpolation is required when there is a gap of more than 15 minutes between two points, but the starting and ending points are below 3000 ft.
3. *EXTR*: Extrapolation is required when the starting or ending points are above 3000 ft, indicating that receivers did not capture the arrival or takeoff. If the metadata has arrival and departure airport information, we use it to extrapolate the trajectory based on scheduled departure or arrival time.
4. *INCP:* Trajectory remains incomplete if it is not possible to extrapolate to complete the trajectory. This occurs when extrapolation is required, but there isn't any information about the departure or arrival airport.
5. *BELOW3K*: Flights are airborne but remain below 3,000 feet. These flights are not modeled using an aircraft performance model because we use the ICAO default Landing and Takeoff (LTO) cycle, which extends from the ground to 3,000 feet.
6. *RCJT*: Flights are rejected because they have fewer than 3 waypoints or were never airborne.

Table 1 lists the results of the classification for all 62 million flights. Roughly half the trajectories are complete and do not require any interpolation or extrapolation. About 12% of flights never reach above 3,000 feet. These flights are likely general aviation flights, incomplete trajectories, or aborted flights. They are not included in the set of flights that are modeled. The 10% of trajectories classified as INTR are interpolated using functions implemented in pycontrails, where they include step climbs when necessary. 16% of flights can successfully be extrapolated to their origin or destination airports, and this is also done using native pycontrails functions. 11% of flights remain incomplete. 0.7% of flights are rejected due to having fewer than 3 waypoints or never being airborne. Flights classified as FULL, INTR, EXTR, and INCP, totaling 87% of the global flight dataset can be modeled.

Table 1 Results of the trajectory classification

Classification	**% of all flights**
FULL	50.76%
BELOW3K	11.92%
INTR	9.61%
EXTR	16.18%
INCP	10.86%
RJCT	0.67%

Secondary trajectory cleaning

The extrapolation of flights purely based on the arrival and departure airports created unrealistic flight trajectories that required further processing. The unrealistic trajectories were primarily created due to incorrect airports in the metadata or an issue with the implementation of step-climbs in pycontrails. The following cleaning was only done for extrapolated trajectories.

1. **Problems due to incorrect metadata**: Determining if the metadata is incorrect required comparing the extrapolated trajectories and the raw trajectory. If the raw trajectory shows the aircraft is already in its climb (descent) phase at the start (end) of the raw trajectory,[1] the extrapolated trajectory should be short. However, if the extrapolated trajectory has a first (last) waypoint that is greater than 130 km away or greater than 20 minutes before (later), we consider the departure (arrival) airport to be incorrect in the metadata. In this case, we extrapolate backwards (forwards) in time from the first (last) three raw waypoints using the calculated direction of travel, rate of climb, and speed, until the aircraft is at or below 3,000 feet in altitude. We then find the closest airport from this new extrapolated trajectory and use it to replace the incorrect departure (arrival) airport metadata.
2. **Problems due to implementation of step-up climbs:** If the aircraft is in the cruise phase at the beginning of the raw trajectory and requires extrapolating more than 2 hours back in time to a departure airport, the default altitude interpolation in pycontrails would place the climb in the middle of the extrapolated segment.[2] To correct this, we place one waypoint at the departure airport and one waypoint at cruise altitude near the departure airport. For the location and time of the intermediate waypoint, we simulate a nominal climb out from the departure airport. The waypoint is placed at cruise altitude between the airport and the first raw waypoint. We use a nominal rate of climb of 12.7 m/s, a nominal ground speed of 130 m/s, to set the latitude, longitude, and time of the intermediate waypoint. The trajectory is then interpolated between these waypoints and the first raw waypoint to give a realistic trajectory.
3. **Trajectory rejections:** We also rejected extrapolated trajectories where most of the trajectory was extrapolated. If the raw trajectory is less than 10% of the full extrapolated trajectory, either in time or distance, the trajectory is rejected. Trajectories that were longer than 20 hours after the secondary cleaning are also rejected.
4. **Unrealistic flight speeds:** A final test for unrealistic speeds is carried out. Flight speeds are considered unrealistic if:
    a. waypoints above 10,000 feet have a speed lower than 70 m/s or greater than 350 m/s, or
    b. waypoints above 10,000 feet have a rate of climb/descent greater than 5,000 ft /min
    c. waypoints below 10,000 feet have a speed higher 300 m/s

    In all three cases, the waypoints with unrealistic speeds are removed and the trajectory is interpolated again. If unrealistic speeds persist, the process is repeated until there are no more unrealistic speeds, or the trajectory ends up with fewer than 3 waypoint which leads to its rejection.

The resulting set of trajectories are considered processed and are used in the analysis.

---

[1] An aircraft is classified as climbing (descending) at the beginning (end) of the raw trajectory if the first (last) three waypoints show an average rate of climb (descent) greater than 6.35 m/s (1250 ft/min) and the altitude of the first (last) waypoint is less than 75% of the altitude range of all raw waypoints for the flight.

[2] This behavior is to simulate stepped climbs when an aircraft is out of ADS-B range for greater than 2 hours, typically over the ocean, and is at a higher altitude at the end than it was at the beginning of the data gap.

Aircraft type information

The metadata contains conflicting aircraft type names and codes. Aircraft types are preferentially identified by $type\_name$, not ac_icao_code, because type names generally offer more information. For example, the $type\_name$ may include terms such as 'winglets' or 'freighter' that differentiate between the aircraft variants. ICAO codes do not capture such details. To improve data quality and furnish as much data as possible for modeling exercises, we perform the following data cleaning steps:

1. Clean up type names (e.g., {"A320-200N", "Airbus A320-200N"} → "Airbus A320-200N")
2. Assign ICAO type codes based on resemblance of the $type\_name$ to ICAO reference data.
3. Harmonize type names for individual aircraft since some aircraft have partially missing or conflicting type names.
4. Assign an ICAO type code that is supported by Base of Aircraft Data (BADA) aircraft performance model.

Each step is elaborated upon in the next few subsections.

Clean up type names

Aircraft type name cleaning focuses on three issues:

1. If multiple names were identical when ignoring case (e.g., "Cessna 172M Skyhawk" and "Cessna 172m Skyhawk"), the name was set to the more common one according to the number of flights.
2. Various manufacturer names are sometimes abbreviated in the raw data and are consistently spelled out in the cleaned data (e.g., "B737-700" -> "Boeing 737-700").
3. Some types are labeled "Various <manufacuter> Airframes" (e.g., "Various Airbus Airframes") or "Various <manufacturer and model> airframes" (e.g., "Various Airbus A320 Airframes"). Because the former can never uniquely identify an aircraft type, we set these entries to NULL. The latter can potentially identify a type, so we keep these values. We differentiate between the former and the latter by the presence of numbers in the type name (i.e., if the type name contains a number, we assume it contains model information).

Assign ICAO codes

The table below lists various data quality issues related to type names and type codes.

group	flights	share
1. Type name and code missing	4365427	7%
2. Only type code missing	14527	0.02%
3. Type name missing	401514	0.65%
4. Multiple type codes per type name	12502605	20%
5. No issues	44713929	72%

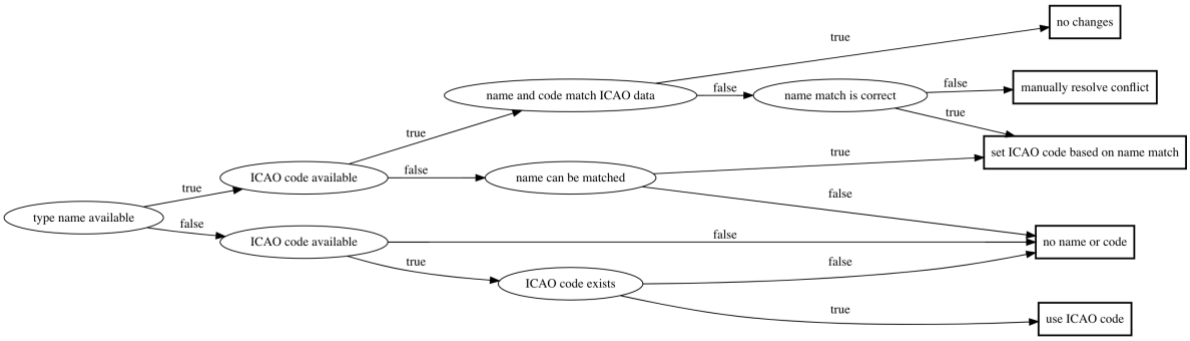
For modeling purposes, we fill as many type codes as possible. To do so, we compare the Spire data with ICAO reference data and make the following assumptions:

1. If the Spire type name and Spire type code resemble ICAO type names and type codes, we assume the name and code is correct.

2. If the Spire type name resembles an ICAO type name but the type codes disagree, we assume that the Spire type code is incorrect and set it to the ICAO type code.
3. If the Spire type name resembles an ICAO type name and the Spire type code is missing, we assume the Spire type code is incorrect and assign the type code from the ICAO data.
4. If the Spire type name is missing but the type code is valid, we assume the Spire type code is correct.

The assumptions use the term "resemble" because Spire type names rarely match ICAO type names. Whether two names resemble each other was determined using Ratcliff/Obershelp pattern recognition with manual corrections in case of conflicting data.

The assumptions are encoded in the following decision tree.



Harmonize types for individual aircraft

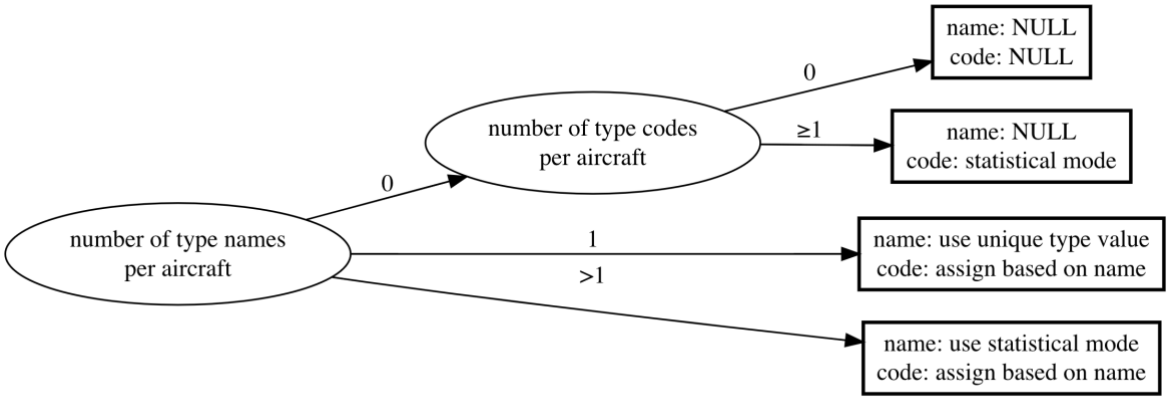
In the raw data, one aircraft can be associated with multiple type names and ICAO codes over time. An individual aircraft is identified by their 24-bit ICAO address associated with their ADS-B transponder. The following table looks at the number of type names and type codes per individual aircraft. 63% of aircraft have no type information and so cannot be modeled, but this accounts for only 4% of flights in the dataset.

Group	Aircraft	Flights	Aircraft Share	Flight Share
One type name	181512	51080966	27%	82%
At least one type name	46224	6924777	7%	11%
At least one type code	20607	1679860	3%	3%
No type name or code	430716	2312399	63%	4%

For modeling purposes, we harmonize type names and codes per aircraft, primarily to fill as many values as possible. To do so, we make the following assumptions:

1. Each aircraft is uniquely identified by one ICAO address
2. Each aircraft is best described by the most common (in terms of number of flights) type name associated with it.
3. In the absence of a type name, each aircraft is best described by the most common (in terms of number of flights) ICAO code associated with it.
4. Each type name is associated with one and only one ICAO code.
5. In the absence of both type name and code, we do not have enough information to model the flight.

These assumptions are encoded in the following decision tree.



The table below shows the impact of harmonizing ICAO type codes per aircraft in terms of the number of flights that have been assigned an ICAO code before/after this correction. At the end of the process 96% of flights have a valid ICAO type code with each aircraft associated with one unique ICAO type code.

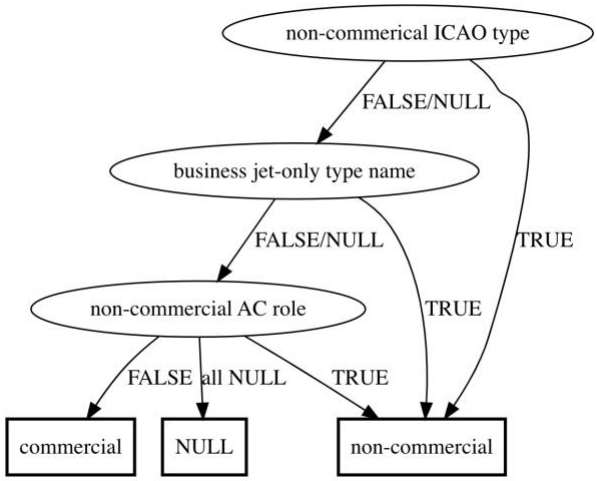
group	flights	share
before	57632284	93%
after	59684770	96%

Defining Commercial Aviation

We define commercial aviation as flights that transport passengers, cargo, or mail in exchange for payment. However, ADS-B data offers no clear way to identify commercial vs. non-commercial flights. It is easier to identify non-commercial flights using heuristics based on aircraft types, aircraft engines etc. So, we use heuristics outlined below to categorize flights as non-commercial aviation. All other flights considered commercial aviation. The following aircraft are categorized as non-commercial flights:

1. Aircraft types that have piston engines.
2. Aircraft types that have special designators.
3. Aircraft types that are solely used as business jets. Aircraft types are listed in Appendix D of Sitompul and Rutherford, 2025)
4. Aircraft with non-commercial aircraft roles as defined by the ac_type field from Spire (e.g., ambulance, hospital, military, …)
5. If none of the preceding information is available, the commercial status is unknown (NULL). This occurs when the type code, type name, and aircraft role are missing

After filtering out non-commercial flights, 37.4 million flights remain in the dataset for further analysis. The decision logic is plotted in the following decision tree



Meteorological data

Description

Data provider: European Centre for Medium-Range Weather Forecasts (ECMWF)
Modeling aircraft fuel use, estimating aircraft performance, and simulating contrail lifetimes require information about the atmosphere in the vicinity of the flight.

The European Centre for Medium-Range Weather Forecasts (ECMWF) provides global meteorological data in the form of its ERA5 HRES data product (cite) (Hersbach et al., 2020). This can be freely downloaded from the ECMWF Copernicus Climate Data Store. The data is downloaded at longitude–latitude grid resolution of 0.25° × 0.25° over 37 pressure levels (sea level to 42,000 ft above sea level) and at a 1-hour time resolution.

Preprocessing

A key variable in determining the formation and lifetime of a contrail is the relative humidity with respect to ice (RH_i) of the atmosphere. ERA5 has been shown to overestimate the RH_i with respect to radiosonde data (cite Agarwal 2022) and underestimate it when compared to in-situ measurements taken by aircraft part of the In-Service Aircraft for a Global Observing System (IAGOS) fleet (cite). In addition, ERA5 data rarely captures the high-supersaturation ($RH_i > 120\%$) that the in-situ sensors measure. Thus, a correction to the RH_i field is required to make it more closely resemble real world data. We use a correction developed by Teoh et al., 2024 that changes ERA5 data such that the probability density of the RH_i more closely resembles that seen by the IAGOS aircraft.

Airport data

Description

Data provider: OurAirports
We considered multiple sources for airport master data. The ICAO API was identified as the ideal source as it is one of the industry standards, but it was deemed prohibitively expensive. In the absence of authoritative data, OurAirports was identified as the next best data source because:

- All required attributes (ICAO code, IATA code, name, country, coordinates) are included, plus they include nice-to-have information (e.g., Wikipedia URL and website URL).
- OurAirports publishes daily updates dating back to November 2021 in their GitHub repository, meaning that we have access to regular snapshots and can track changes over time.
- The dataset is free and open.
- Their coverage of airports was better than competitors (e.g., IP2Location).
- The Python library for modeling contrails pycontrails also uses OurAirports.

Preprocessing

A key challenge with OurAirports is that the data is essentially crowdsourced and not authoritative. Users can add to and edit airport data, which means that data quality and coverage generally improve over time. It is therefore difficult to differentiate between corrections in the data and genuine changes in the underlying airports. This distinction is important because airport codes, names, locations, etc. can change over time, and we require accurate historical data for our analyses.

We store monthly snapshots of OurAirports data dating back to November 2021. OurAirports assigns a unique identifier to each airport. This identifier is used to track changes over time. Cleaning all historical data would be prohibitively time consuming. Instead, we generally regard each airport's latest record as accurate and flag if historical values agree or disagree with the latest value. We undertake basic cleaning of countries, IATA codes, and ICAO codes where we resolve conflicting data entries, cases where the same code is associated with multiple airports, and cases where the same airport is associated with multiple codes. These cleaning processes do not affect the emissions modeling but can affect the attribution of emissions to countries.

Airline fleet and engine data

Description

Data provider: Spire, IBA, ICAO Engine Emissions Databank (EEDB), European Environment Agency (EEA), United States Environmental Protection Agency (US EPA), Swiss Federal Office of Civil Aviation (FOCA)

1. Spire
   The Spire data provides flight-level information, including aircraft registration and type. By using this information, we can identify the aircraft engine type by cross-referencing with the IBA dataset.

2. IBA
   The IBA data provides information on registered aircraft fleets for each airline, including engine types. To merge this with the Spire data, we use the 'aircraft model' and 'tail number' columns. This allows us to determine the engine type associated with each specific flight.

3. ICAO EEDB
   The ICAO Engine Emissions Databank (ICAO EEDB) is used for gaseous emissions, smoke, and nvPM emissions certification measurements for turbofan engines. To merge EEDB with flight-level engine data, we use the 'Engine Identification' column, and the 'Engine UID' serves to link each engine to its unique emissions testing results.

4. EEA
   For turboprop and piston engines, the EEA's aviation emissions inventory was used to match aircraft types with the relevant LTO and cruise phase emissions for NOx, CO, SOx, and water vapor.

6. US EPA
   For turboprop and piston engines that were not available within the EEA dataset, constant values for nvPM emissions were used from the US EPA air quality manual (cite EPA 2020).

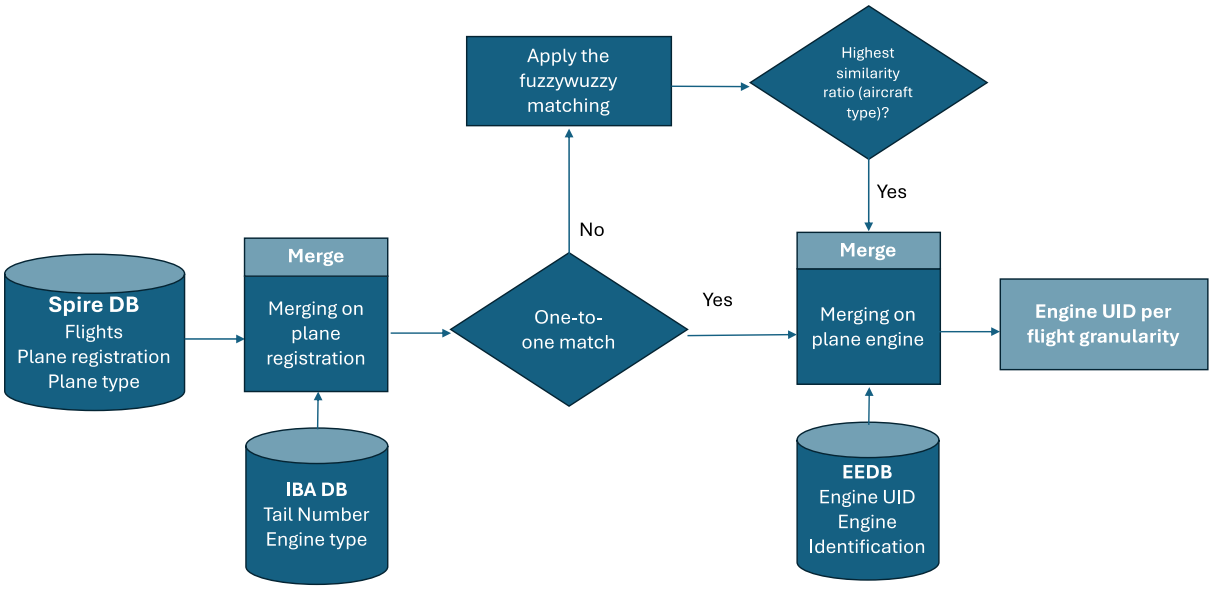
7. Swiss FOCA
   For piston engines that were not available in the EEA dataset, default values from the Swiss FOCA Aircraft Piston Engines Summary Report (cite FOCA 2007).

Details of the cruise and LTO emissions modeling is provided in the Cruise emissions modeling and Landing and takeoff emissions modeling sections.

Preprocessing

The process of assigning airline fleets and engine types begins with flight-level data from the Spire database. Using the $tail\_num$ column from Spire, we merge it with the tail number column in the IBA database. However, since aircraft tail numbers can have special characters and spaces, not all aircraft tail numbers have a one-to-one match in the IBA data. We use the Ratcliff/Obershelp pattern recognition as implemented in the FuzzyWuzzy Python library to identify the closest matching aircraft types and improve the accuracy of the merger.

For flights with duplicate aircraft registrations, we select the aircraft type with the highest similarity score, representing the closest character match. This step allows us to assign an engine type at the flight level. Finally, we merge this engine type with the EEDB to retrieve the corresponding Engine UID.



The EEDB only covers jet engines and does not have any information about turboprop aircraft. Jet engines account for 52% of the entire trajectory dataset or 87% of commercial flights. Of the jet engine aircraft, we get exact matches on tail number for 93% of them and we can assign

specific engines UIDs to them. For the rest, a default engine is chosen. The default engines are stored in the pycontrails GitHub repository.

Operational data

Description

Data provider: OAG, US DOT, ICAO, ch-aviation
The OAG dataset provides flight schedule data for commercial flights. The dataset we use does not have *per flight* schedule data, rather it is grouped by departure airport (origin), arrival airport (destination), aircraft type, and airline with a frequency column that represents the number of annual flights for that combination. The OAG dataset forms the basis of our inventory by defining the airport pairs, airline, aircraft, and frequency of flights for 2023.

From the US DOT, we use T100 data to get load factor information for flights arriving and departing the US. From ICAO we use the Traffic by Flight Stage (TFS) dataset to assign the passenger load factor and the weight of belly cargo to all international flights. We use airline-specific load factors from ch-aviation as well.

Preprocessing

For US departing and arriving flights, the T100 dataset is used to assign a load factor and a weight of onboard freight and mail from T100 dataset to each flight's departure airport, arrival airport, specific aircraft type ,and air carrier combination in OAG. The matching steps and granularity are as follows:

- The OAG dataset is merged with the T100 dataset by the departure airport, arrival airport, T100 aircraft name and air carrier, to calculate the load factor for each combination. For unmatched flights, the table below shows the fallback matching combinations and granularities:

Granularity	**Variables used for matching**
Exact match	Airport pair, aircraft type and carrier
Fallback 1	Airport pair and carrier
Fallback 2	Airport pair

- The average weight of onboard freight and mail for the same route and aircraft class is assigned to all combinations that were not successfully matched.

For international flights, the ICAO Traffic by Flight Stage (TFS) dataset is used to assign the passenger load factor and the weight of belly cargo to each flight's departure city, arrival city, specific aircraft type, and air carrier combination in the OAG database. For combinations that were not matched, fallback data with different levels of granularity was used. Freighter and passenger flights were separated into two data frames for merging,
The passenger flights:

- Passenger count
  - The table below shows the various the variables used to match at each level:

Granularity	**Variables used for matching**
Exact match	City pair, aircraft type and carrier
Fallback 1	City pair, aircraft class
Fallback 2	Distance bin, aircraft class

  - Air carrier average passenger load factors from Ch-aviation were used for all the unmatched combinations and a global fallback passenger load factor of 82.4% was used.

    - For non-US domestic flights that the ICAO dataset doesn't cover, air carrier average load factors from Ch-aviation were used to calculate passenger payload and revenue passenger traffic.
- Belly cargo
    - The belly cargo mass is calculated by attaining the cargo mass fraction (CMF) using the following formula:

$$CMF = \frac{\text{freight revenue traffic} + \text{mail revenue traffic}}{\text{passenger revenue traffic} + \text{freight revenue traffic} + \text{mail revenue traffic}}$$

$$Belly\ cargo\ mass = CMF * \frac{\text{calculated passenger count} * 100kg}{1 - CMF}$$

    - The table below shows the variables used for each level

Granularity	**Variables used for matching**
Exact match	City pair, aircraft type and carrier
Fallback 1	City pair, aircraft class
Fallback 2	Distance bin, aircraft class

The dedicated freighters:

- Annual total payload capacity and payload factor from the ICAO TFS freighter dataset is matched to the dedicated freighters in the OAG dataset using aircraft type and carrier. The table below shows the fallback matching and the variables used for matching at each stage:

Granularity	**Variables used for matching**
Direct match	Aircraft type and carrier
Fallback 1	Aircraft type
Fallback 2	Single fallback 44%

Schedule data from Spire was used for UPS and FedEx carriers, which the OAG dataset does not include. The frequency of each flight was determined by calculating the number of occurrences for each combination when grouping the datasets by the departure airport, arrival airport, carrier, and aircraft type.

For non-US combinations, UPS and FedEx carrier combinations were selected from the ICAO dataset, and the median value of the average payload capacity and weight occupancy factor was calculated for the first round of merging based on aircraft type and carrier. For unmatched combinations, the median average payload capacity and weight occupancy factor was used.

Similarly, for UPS and FedEX flights operating in the US, T100 dataset was used for payload factors. The T100 UPS and FedEx combinations were grouped by route, carrier, and aircraft type to calculate the median value of the average onboard freight and mail (tons). These values are used for any direct matches. For the unmatched combinations, the carrier and aircraft type median values were used.

Models

Aircraft performance model

Model: BADA3

BADA is a collection of aircraft performance models developed and maintained by EUROCONTROL. It provides standardized performance parameters and fuel consumption models (objective functions) for a broad range of commercial aircraft, enabling estimates of aircraft behavior across different operational scenarios. BADA 3 is one of the most widely used versions, but not the latest and most accurate. It combines aerodynamic data, thrust models, and phase-specific fuel consumption formulas to deliver performance simulations outputs across a variety of aircraft types. For the aircraft types that have not been modeled, they use synonym aircraft that are very similar in weight, performance and mission.

Cruise emissions modeling

CO_2, SOx, and H_2O emissions

CO_2, SOx, and H_2O emissions are calculated using constant emission indices on the amount of fuel burned.

Model	Emission factors	Percentage of commercial flights in the database	Reference
Constant emissions index	3.84 kg kg−1 for CO_2 1.237 g kg−1 for H_2O 1.2 g kg−1 for SOx	100%	ICAO, 2022 Wilkerson et al., 2011 Lee et al. 2021

NOx, HC, and CO emissions

Model: Fuel Flow Method 2
The Fuel Flow Method 2 is a way to adjust emissions data from the ICAO engine emissions database, which relies of ground testing of engines, to estimate NOx, HC, and CO emissions at altitude (cite).

For any engines that are not represented in the ICAO Engine Emissions Database, we use constant emission factors to estimate emissions.

Model	Emission factors	Percentage of flights in the database	Reference
Fuel Flow Method 2	Dependent on engine operating conditions	87%	DuBois and Paynter, 2006
Constant emissions index	15.14 g kg−1 for NOx 3.61 g kg−1 for CO 0.520 g kg−1 for HC	13%	Lee et al. 2021

nvPM Emissions

Model: T4/T2 method

The T4/T2 method is used to estimate cruise nvPM emissions profiles, which interpolates EEDB nvPM certification points using the ratio of turbine-inlet temperature (T4) to compressor-inlet

temperature (T2) (cite Teoh et al.). For any engines that are not represented within the EEDB, we use constant values for nvPM number and nvPM mass.

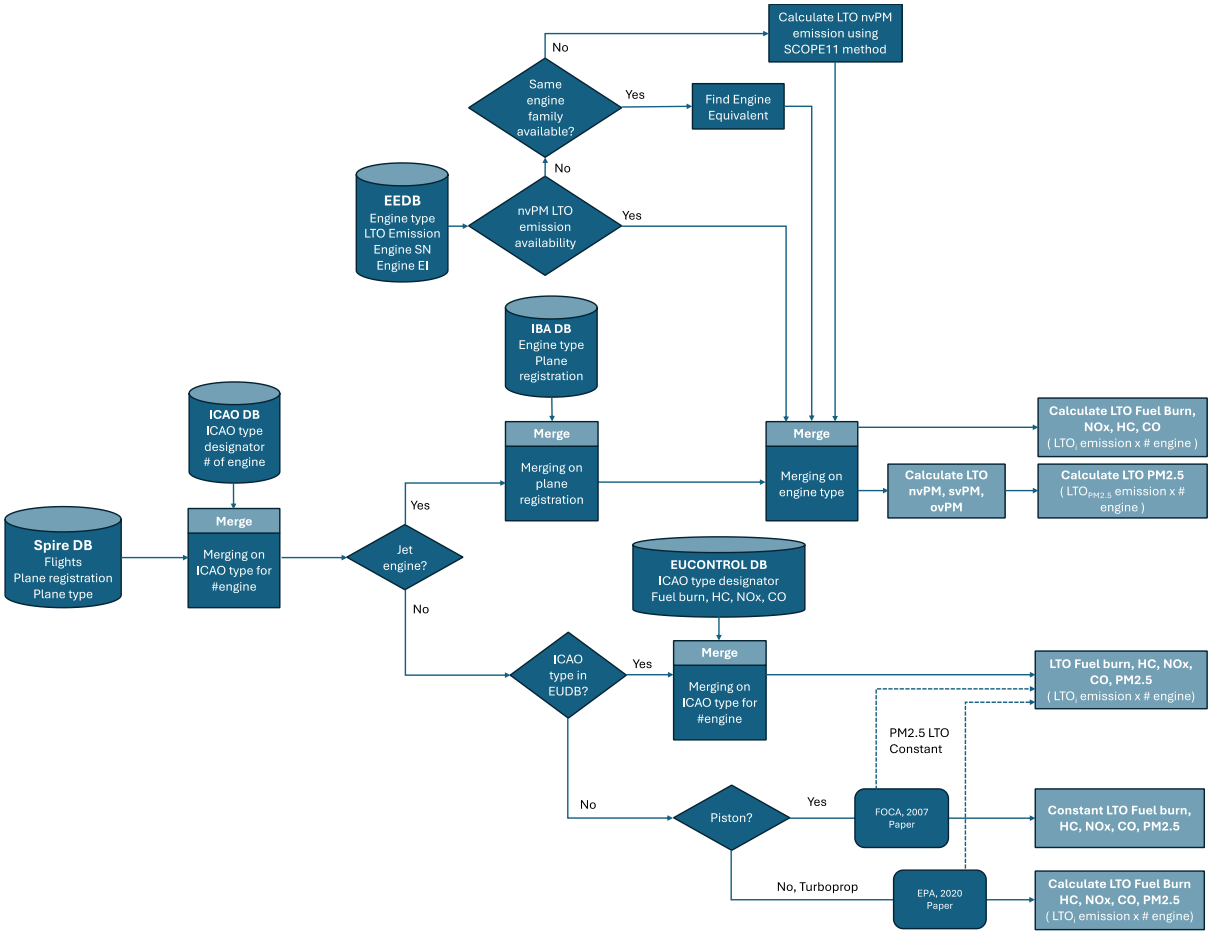
Model	Emission factors	Percentage of flights in the database	Reference
T4/T2 method	Dependent on engine operating conditions	87%	Teoh et al. 2022
Constant emissions index	0.088 g kg−1 for nvPM mass 10^{15} kg^{-1} for nvPM number	13%	Stettler et al. 2013

Landing and takeoff emissions modeling

Model: SCOPE11 and EPA
Data:

- **Engine Emission Databank** (Source link)
- **EUROCONTROL** (Source link)
- **EPA**
- **FOCA**

To model landing and takeoff (LTO) emissions, we use several datasets depending on the engine type. As shown in the figure below, each engine type has its own model and data source for LTO emission calculations.



The modeling hereby, discuss per engine type basis:

1. Jet Engine

For jet engine aircraft we use the EEDB to calculate the total LTO emission based on the engine type. The EEDB shares the fuel burn, CO, NOx, HC, and nvPM of several engine. Since the nvPM sheet only includes 130+ unique engines, there are several methods needed as a fallback as shown in the table below.

Granularity	Data	Variable description
Direct match	EEDB LTO emission	Matching using Engine Identification
Fallback 1	EEDB LTO emission	Matching Engine Identification from the same family (different generation)
Fallback 2	EEDB LTO emission	Matching Engine Identification with the same range of thrust
Fallback 3	EEDB Smoke Number	Engine that contributes < 0.1% of flights and no match with direct match and fallback 1

For fallback 3 we use the SCOPE11 method to calculate some engine (< 1% flights) using SN for engines that are not available at in the EEDB.

SCOPE11 Method

The SCOPE11 method uses non-volatile particulate matter (nvPM) emission data as input. Before CAEP10, only the particulate-related standard was based on the Smoke Number (SN). For some engines, nvPM emission data are not available in the Engine Emissions Data Bank (EEDB), so only SN measurements can be used to estimate nvPM emissions. In Step 0, calculations are performed sequentially, using the SN to derive nvPM concentration (CImass), which is then used to compute the nvPM emission index in terms of mass (EI_{mass}). To derive the nvPM concentration (CI_{mass}), the following equation is used:

$$\mathrm{CI_{mass}}\left[\frac{\mu g}{m^3}\right] = \frac{648.4 * e^{(0.0766 * SN)}}{1 + e^{(1.099 * (SN - 3.064))}}$$

The emission index (EI) is calculated by multiplying the concentration (CI_{mass}) by the volumetric flow rate Q_{mode} (measured in m^3/kg). The flow rate Q_{mode} depends on the Air-to-Fuel Ratio (AFR) and the reference by-pass ratio β, given by:

$$\mathrm{Q_{mode}}\left(\frac{m^3}{kg}\right) = 0.776 * (AFR) * (1 + \beta) + 0.767$$

The AFR at each of the four LTO points have been estimated by Wayson et all. As 106 at Idle, 83 at Approach, 51 at Climb-Out and 45 at Take-Off.

A correction factor k_{slm} is used to adjust for measurement system losses. It depends on the reference by-pass ratio β and nvPM concentration (CI_{mass}), calculated as:

$$\mathrm{k_{slm}} = \ln\left(\frac{3.219 * \mathrm{CI_{mass}} * (1 + \beta) + 312.5}{\mathrm{CI_{mass}} * (1 + \beta) + 42.6}\right)$$

The nvPM emission index by mass (EI_{mass}) is derived for each of the four LTO modes as:

$$EI_{mass}\left[\frac{mg}{kg}\right] = CI_{mass} \cdot Q_{mode} \cdot k_{slm}$$

For the nvPM emission index by number (EInum), the mean particle size across all modes is used, with the number density calculated by:

$$EI_{num}\left[\frac{particles}{kg}\right] = \frac{EI_{mass}}{\left(\frac{\pi}{6} * 10^9 * \left(\frac{GMD}{10^9}\right)^3 * e^{(4.5 * (\ln(1.8))^2)}\right)}$$

This method allows for a step-by-step derivation of EImass and EInum from the Smoke Number, enabling estimation of nvPM emissions for engines lacking direct nvPM emission data in the EEDB.

After getting the nvPM of each engine LTO, to get the PM2.5 we calculate it using EPA procedure below:

EPA Procedures for Emission Inventory Preparation

To estimate the total PM2.5 emissions from each jet engine aircraft, we use EPA procedures to calculate using the time in mode for each of the four components of the LTO cycle can be found in the EPA Procedures for Emission Inventory Preparation document (Environmental Protection Agency, 1992):

$$fuel\ in\ mode\ [kg] = fuel\ flow\ [kg/s] * TIM\ [min] * 60\ s/min$$

Table below shows the Time in Mode (TIM) for every mode of flights.

Table. Time in mode assumed for air pollution modeling

Mode	TIM (m)

T/O	0.7 mins
Climb	2.2 mins
Approach	4.0 mins
Idle (Taxi)	26 mins

Source: EPA, Procedures for Emission Inventory Preparation

svPM emission results when fuel containing sulfur is combusted in an aircraft engine. It is directly proportional to the sulfur content of jet fuel. ICAO's Air Quality manual suggests a conversion efficiency of 0.024 (m/m) of sulfur to vPM and sulfur content of 680 ppm (0.068% -- see ICAO Air Quality Manual Sec 3.8). Accordingly:

$$LTO\ svPM\ [mg] = 0.024 \times 0.068\% \times LTO\ fuel\ [kg] \times 1{,}000{,}000\ mg/kg \times \mathrm{NE}$$

Where:
LTO svPM = total svPM emissions during landing and take-off
LTO fuel = total fuel during the Landing and Take Off
NE = Number of engines on the aircraft

ovPM emission occurs when unburnt hydrocarbons from the fuel adsorb onto a fine particle, adding to its mass. ovPM is largely untested across EEDB engines and so is calculated using the CFM56-2-C1 as a reference (see Sec. 3.9). It is assumed to be directly proportional to LTO fuel for all engines in the EEDB:

$$LTO\ ovPM\ [mg] = (4.6 \times TO\ fuel\ [kg] + 3.8 \times CO\ fuel\ [kg] + 4.5 \times App\ fuel\ [kg] + 11.3 \times Idle\ fuel\ [kg]) \times \mathrm{NE}$$

Where:
LTO ovPM = total ovPM emissions during landing and take off
TO fuel = kilograms of fuel burn during takeoff
CO fuel = kilograms of fuel burn during climb out
App fuel = kilograms of fuel burn during approach
Idle fuel = kilograms of fuel burn during idle/taxi
NE = Number of engines on the aircraft

The equation to calculate the final $PM_{2.5}$ value is as follow:

$$PM2.5\ [g] = nvPM[g] + svPM\ [g] + ovPM\ [g]$$

Where:
nvPM = represents non-volatile particulate matter,
svPM = denotes sulfate particulate matter, and
ovPM = refers to organic particulate matter.

2. Turboprop and Piston

We use the EEA emissions calculator to get the LTO emissions for each aircraft using the default ICAO value. The database does not include the PM2.5 value of turboprop engines, so we get the constant of 1.234 pounds per LTO that is taken from the EPA document (EPA, 1992 – Procedures for Emission Inventory Prep).

For other flights with turboprop engine that aren't available in the EEA dataset, we get the value constants from EPA document, 2020 National Emission Inventory: Aviation Component. The LTO fuel burn value is taken from the weighted mean of the LTO fuel burn value of all flights with turboprop engines that are available in the Eurocontrol database (80% of the total turboprop flights in the 2023 Spire inventory data).

For other flights with piston engine that aren't available in the EEA database, we get the value constants from Swiss Federal Office of Civil Aviation (FOCA) document (Aircraft Piston Engine Emissions Summary Report).

With this all accounted, the final fallback constant is shown below:

Engine Type	Fuel LTO Cycle (kg)	HC LTO Total mass (g)	CO LTO Total Mass (g)	NOx LTO Total mass (g)	PM2.5 LTO Emission (mg)
Piston	7.53	174	7327	24	521.46
Turboprop	120.984	77.11	12759.54	72.57	

The resulting calculation using the Spire database:

Engine Type	Category	# flights	% of Engine Type
Jet	Same Engine Name	30,257,617	93.1%
	Equivalent Engine	2,038,781	6.3%
	SN Generated (<10,000 flights per engine)	187,633	0.6%
Turboprops	Eurocontrol	4,974,220	80%
	Constant (EPA & ICAO)	1,232,112	20%

Analysis of results

Merging with airline schedule data

ADS-B transmitters are placed on most aircraft, and there is no easy way to classify the flights as commercial or non-commercial. However, commercial flights are represented in flight scheduling data. We use OAG's flight schedules as a database of commercial flights and map the results of the emissions inventory onto the OAG schedule to represent the emissions of commercial aviation. OAG schedules do not include operations from FedEx and UPS. For those operators, we rely purely on the trajectory data.

The mapping between OAG and emissions inventory is not perfect, and there are multiple issues to solve

1. The OAG data does not have each individual flight, rather, it is grouped by departure airport (origin), arrival airport (destination), aircraft type, and airline, with a frequency

column that represents the number of flights for that combination. So, in matching with the emissions inventory, which is per-flight, multiple flights from the inventory must be merged onto each row of the OAG database.
2. OAG data uses IATA codes for airports and aircraft while Spire (which is the basis of the emissions inventory) uses ICAO codes for airports and aircraft. There is a many-to-many relationship between the ICAO and IATA codes for aircraft. Not every airport has an ICAO code and not every airport has an IATA code.
3. Since airport and airline information is not transmitted directly over ADS-B receivers, there can be flights that have trajectory information but do not have airport or airline information.

These data issues necessitate the use of fallbacks to ensure that every entry in the OAG database can be associated with an appropriate fuel consumption. We use 4 levels of matching for this.

1. Level 1: Flights in the emissions inventory that have an exact match with rows in the OAG data on origin, destination, aircraft type and airline are associated with the corresponding rows. The number of flights from the emissions inventory that are matched may not correspond to the frequency number in the OAG data. To account for the discrepancy the emissions are scaled to the frequency reported in the OAG data. The unmatched OAG data is used at the next step.
2. Level 2: For the remaining OAG data, flights in the emissions inventory that match on origin, destination, and aircraft type are used to calculate the emissions. The total emissions of all the matched flights are scaled to the frequency reported in the OAG data. The unmatched OAG data is used at the next step.
3. Level 3: For the remaining OAG data, the aircraft type and the distance between the origin and destination airport is used to estimate the fuel burn. To do this calculation, all the previously matched emissions inventory data is used to model a linear relationship of fuel burn to distance for each aircraft type. This model is queried at the distance between the origin and destination airport to get emissions per flight. The emissions per flight are scaled by the frequency reported in the OAG data to get the total emissions. The only OAG data that remains unmatched are those that have aircraft types that are not represented in the emissions inventory.
4. Level 4: For the remaining OAG data, we use the EEA emissions modeling tool. This tool uses the aircraft type and distance to provide an estimate for a flight's emissions. These emission values are scaled by the frequency reported in OAG to get the total emissions.

The table below shows the number of flights in the OAG data that were matched at each stage and the fuel burn total after each stage of matching.

	Matching variables	Number of flights matched (million)	Fuel burn total (Mt)
Level 1	Origin, destination, aircraft type, and airline	28.6 (79%)	226.4
Level 2	Origin, destination, and aircraft type	31.8 (87 %)	242.9
Level 3	Aircraft type	36.1 (99%)	269.2
Level 4	No match; Fuel burn and emissions scaled by distance	36.4 (100%)	269.3

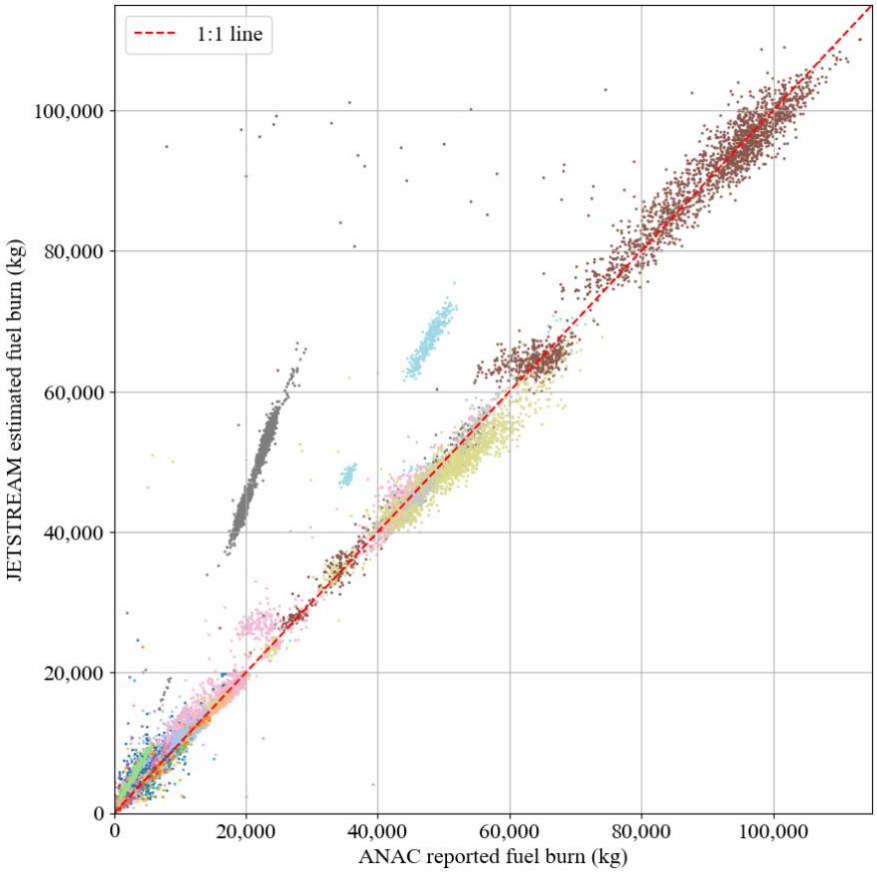
Validation of results

We conduct flight-level and region pair-level validation of the JETSTREAM CO_2 and fuel burn results coming from BADA3 against real-world datasets. In addition to flight-specific and region-pair-specific comparisons, the bottom-up 2023 fuel usage estimate can be compared to the top-down estimate for fuel usage by commercial aviation from IATA. The total fuel usage as estimated by the JETSTREAM inventory, which has been projected onto commercial operations, is 269.3 Mt (88 billion gallons). This is 4.4% lower than IATA's top-down estimates of 92 billion gallons.

Flight-level comparisons to Brazilian airline emissions

We validate JETSTREAM against 427,379 flights with reported fuel burn from Brazil's Agência Nacional de Aviação Civil (ANAC) at the flight level. Matching between the ANAC dataset and JETSTREAM is performed by merging on route, aircraft, type, and departure/arrival time (within a 20 minute window). ANAC data with fuel burn less than 100kg is discarded as misreported. On average, ANAC takeoff times lead Spire takeoff times by about 10 minutes and lag Spire landing times by about 5 minutes. We find ANAC fuel burn is highly correlated with JETSTREAM fuel burn (R^2=0.95). The median of the error between the two datasets is 9.2%.

The plot below plots each flight as an individual point on a plot where the x-axis is the fuel burn reported to ANAC and the y-axis is the estimated fuel burn from JETSTREAM. The red dashed line represents a slope of 1. The closer a point is to this red dashed line, the lower the error in the fuel burn estimate. Except for a few aircraft types, there is good agreement between the modeled fuel burn estimates from JETSTREAM and the reported fuel burn. Removing these aircraft from the validation dataset reduces the median of the per flight error from 9.2% to 8.0%.



For the total fuel burn estimate across all flights considered here, JETSTREAM overestimates it by 9.8% when considering all aircraft, and by 5.4% when the problematic aircraft are excluded.

Region-pair comparisons to CORSIA reporting

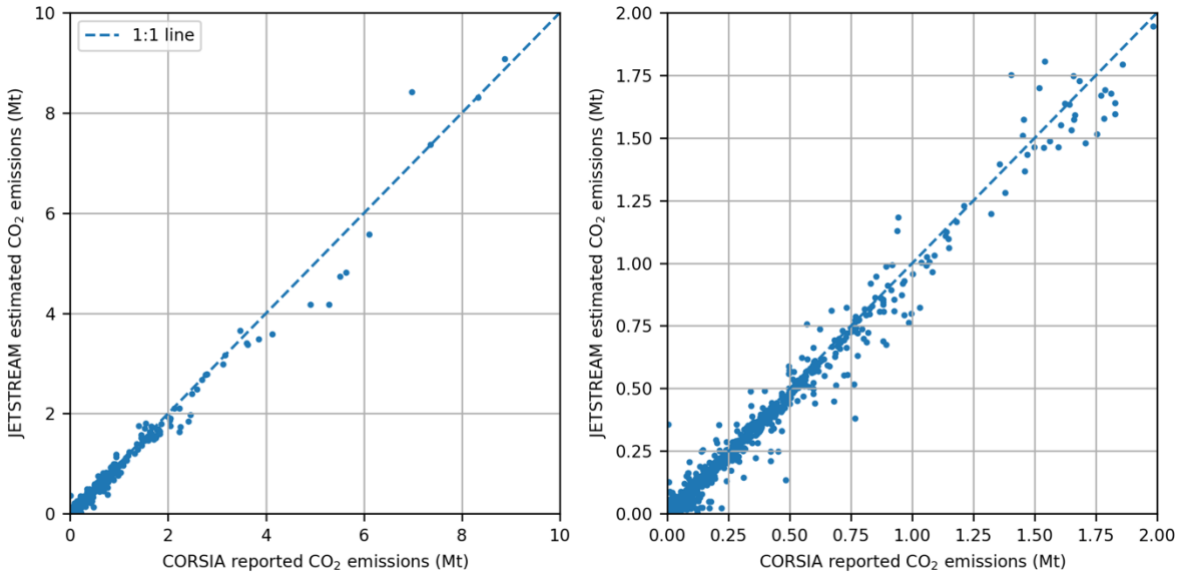
The CORSIA (Carbon Offsetting and Reduction Scheme for International Aviation) is a global initiative by ICAO to track and reduce CO_2 emissions. Data from CORSIA include a mixture of reported and modeled CO_2 totals by region pairs for international aviation. The CO_2 emissions reported to CORSIA are tank-to-wake using an emission factor of 3.16 kg of CO_2 per kg of fuel use. We use the same emission factor to convert the mass of fuel burned calculated in the emissions inventory to a comparable CO_2 emissions number. We use the JETSTREAM inventory that has been projected onto commercial operations schedules for this comparison.

There are two data issues in this comparison. First, some CORSIA data is considered classified, so there are some underestimated or missing values at the region pair level; however, global reported international aviation CO_2 values include the contribution from these classified CO_2 data. Second, the CORSIA regions definition are coarse and do not specify which airports are included in each region pair, so there is uncertainty in what airports—and what individual flights—are considered. We use region definitions from the OurAirports database and use our best knowledge of the choices CORSIA made about assigning disputed territories to regions.

At the highest level, we compare the total CO_2 emissions reported to CORSIA to the total emissions from all international flights in the JETSTREAM inventory. JETSTREAM underestimates the total emissions from international flights by 4.9%. States reported a total of 530 Mt of CO_2 emissions from international flights in 2023 whereas we calculate 505 Mt of CO_2 emissions from international flights.

We compare the 2023 CO_2 CORSIA data against JETSTREAM by aligning region pair definitions from JETSTREAM to our best knowledge of the choices CORSIA made about disputed territories and by filtering to only international flights. CORSIA contains 8,952 unique state pairs. 901 state pairs had some data that was considered confidential, of which 779 pairs had no data at all, while the other 122 had some subset of their emissions available. The JETSTREAM dataset includes 5,658 unique state pairs with emissions information for each pair. There are 3,587 state pairs that are represented in CORSIA, but not in the JETSTREAM inventory. Conversely, there are 554 state pairs that are represented in the JETSTREAM inventory but are either not in the CORSIA inventory or their data is completely confidential.

When comparing at the region-pair level, we only use region pairs that are present in both datasets and have complete data (no confidential data) in CORSIA. The plot below shows each region pair on a plot where the x-axis is the CO_2 emissions reported to CORSIA, while the y-axis is the emissions estimate from JETSTREAM. The plot on the left shows all region pairs, while the plot on the right zooms into region pairs with less than 2 Mt of CO_2 emissions between them. The dashed line represents the 1:1 trend, which is the ideal result. Points further away from the dashed line have larger errors. The error is defined as the percentage difference between the JETSTREAM estimate and the CORSIA reported value. JETSTREAM tends to underestimate CORSIA, with a median error -3.7% across all region pairs.


1:1 line
JETSTREAM estimated CO_2 emissions (Mt)
CORSIA reported CO_2 emissions (Mt)
0
2
4
6
8
10
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00