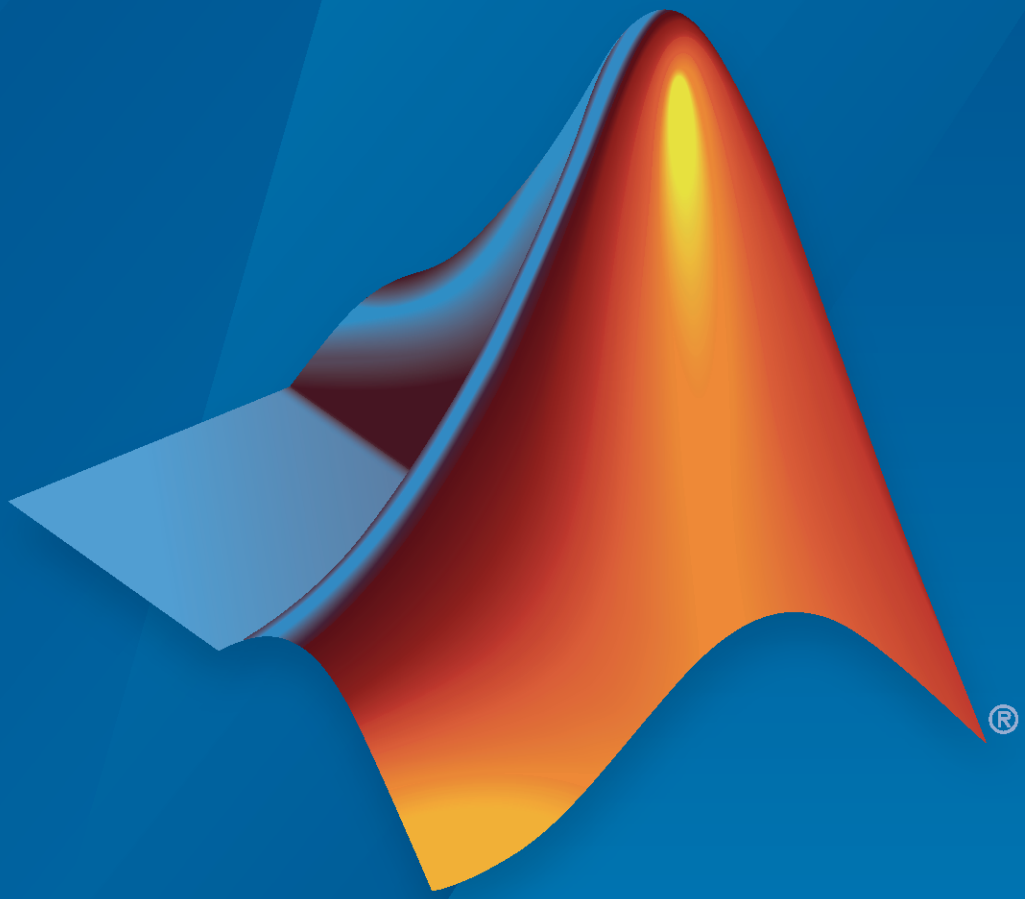
Aerospace Toolbox Release Notes



MATLAB®

MathWorks®

How to Contact MathWorks

Latest news:	`www.mathworks.com`
Sales and services:	`www.mathworks.com/sales_and_services`
User community:	`www.mathworks.com/matlabcentral`
Technical support:	`www.mathworks.com/support/contact_us`

Phone: 508-647-7000

The MathWorks, Inc.
1 Apple Hill Drive
Natick, MA 01760-2098

Aerospace Toolbox Release Notes

© COPYRIGHT 2006–2022 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See `www.mathworks.com/trademarks` for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

Patents

MathWorks products are protected by one or more U.S. patents. Please see `www.mathworks.com/patents` for more information.

Contents

R2022a

R2021b

R2021a

R2020b

R2020a

R2019b

R2019a

R2018b

R2018a

R2017b

R2017a

R2016b

R2016a

R2015b

R2015a

R2014b

R2014a

R2013b

R2013a

R2012b

R2012a

R2011b

R2011a

R2010b

R2010a

R2009b

R2009a

R2008b

R2008a

R2007b

R2007a

R2006b

R2022a

Version: 4.2

New Features

Bug Fixes

New Axes Transformation Function

To convert body frame to stability frame transformation matrices, use the `dcmbody2stability` function.

Improvements to Aero.FixedWing Objects

- The `forcesAndMoments` and `staticStability` methods of the `Aero.FixedWing` class have a new `OutputReferenceFrame` property to specify output references.
- The `Aero.FixedWing.State` class has these changes:
  - To transform body axes to stability axes, use the `BodyToStabilityMatrix` property.
  - To transform stability axes to body axes, use the `StabilityToBodyMatrix` property.

Satellite scenario enhancements

- Create a Walker-Delta constellation in a satellite scenario

  Use the new `walkerDelta` object function.
- Specify custom orientations for satellite and gimbal objects

  The satellite and gimbal object `pointAt` function now supports custom orientations. For an example, see “Modeling Custom Satellite Attitude and Gimbal Steering”.
- Support for loop simulation in satellite scenario

  Manually step through a satellite scenario simulation and modify the asset parameters in the middle of a satellite scenario simulation.
  - `satelliteScenario` now supports these new properties — `SimulationTime`, `SimulationStatus`, and `AutoSimulate`, and two new object functions — `advance` and `restart`.
  - `ConicalSensor` and `Gimbal` objects now support the `aer` function.
- GPS orbit propagators and SEM almanac import

  Satellite scenario now supports system effectiveness model (SEM) almanacs. Use the GPS orbit propagator to propagate an orbit or calculate satellite position.
- Vectorization of satellite scenario setup and analysis

  You can now vectorize the satellite scenario setup workflow and the analysis workflow to gain performance. You can add multiple conical sensors, gimbals, and accesses in a single line of code.
- Non-cluttered satellite scenario constellation visualization

  Create a design using the `ShowDetails` property of `satelliteScenarioViewer` to visualize the constellation in an easy-to-read, non-cluttered manner.
- Multi-hop path selection through large satellite constellation

  The “Multi-Hop Path Selection Through Large Satellite Constellation” example determines the path through a large constellation consisting of 1000 low-Earth orbit satellites to gain access between two ground stations, and also, demonstrates how to calculate the intervals during the next three hour period when this path can be used.

- MATLAB® Compiler™ now supports all satellite scenario generation capabilities.

wait method added to animation objects

These animation objects have a new `wait` method. This method blocks interactions from the MATLAB command line and waits for the animation to stop running before modifying the object.

- `Aero.Animation`
- `Aero.VirtualRealityAnimation`
- `Aero.FlightGearAnimation`

Support for datetime array in ecef2eci, eci2ecef, dcmeci2ecef, decyear, juliandate, and mjuliandate

The `ecef2eci`, `eci2ecef`, `dcmeci2ecef`, `decyear`, `juliandate`, and `mjuliandate` functions now accept `datetime` arrays for the `utc` parameter. Previously, these functions accepted `utc` values as arrays or matrices.

Support for timetable Array in Aero Animation Objects

The `Aero.Animation`, `Aero.VirtualRealityAnimation`, and `Aero.FlightGearAnimation` objects now accept `timetable` arrays in addition to `timeseries` objects. Previously, these objects accepted only `timeseries` arrays.

R2021b

Construct boundary line for visualization

To help you visualize handling and flying qualities, use the `boundaryline` function. This function draws boundary line plots, enabling the representation and identification of quality data along physical boundaries. You can:

- Draw hatched lines representing physical boundaries of system design.
- Qualify data to be within, along, and outside boundaries.
- Control complex boundary curvature and hatched line behavior.

New short-period frequency plot functions for MIL-F-8785C

To help you design robust controllers for aircraft and ensure that aircraft conforms to the MIL-F-8785C standard, use one of the variations of the `shortPeriodCategoryPlot` function: `shortPeriodCategoryAPlot`, `shortPeriodCategoryBPlot`, and `shortPeriodCategoryCPlot`. These functions reproduce the short-period frequency requirements for the category A, B, and C flight phases. You can:

- Display 8785C short-period requirement bounds for Category A, B, and C flight phases and Levels 1, 2, and 3 objectives.
- Overlay short-period frequency and nalpha data to visually verify that 8785C requirements are met.
- Export plots to certification documents to provide key evidence of verification to aircraft certification bodies.

New function to draw altitude envelope contour plot

To help you design aircraft altitude envelopes, use the `altitudeEnvelopeContour` function. An altitude envelope encompasses aircraft design with regards to maximum airspeeds, altitudes, and stall lines for different load factors. You can:

- Represent load factor contours within an altitude-airspeed envelope.
- Display the load factor contours of an aircraft.
- Show physical altitude and airspeed limits.
- Indicate load factor contours with respect to the altitude-airspeed envelope.

New functions to define fixed-wing aircraft

Easily define fixed-wing aircraft. With these functions, you do not need to interact directly with the fixed-wing aircraft objects.

New Function	Class
`fixedWingAircraft`	Create fixed-wing aircraft
`fixedWingState`	Define fixed-wing aircraft state at time instant
`fixedWingCoefficient`	Define numeric coefficients of fixed-wing aircraft
`fixedWingSurface`	Define aerodynamic or control surface on fixed-wing aircraft

New Function	Class
`fixedWingThrust`	Define thrust vector on fixed-wing aircraft
`aircraftEnvironment`	Create aircraft environment
`aircraftProperties`	Create properties defining and managing aircraft

atmoshwm function updated

The `atmoshwm` function has been updated to better handle day and altitude values.

Decimal day values

The `atmoshwm` function now accepts decimal day values. In previous releases, this function floored day values, ignoring partial days.

Altitude values exceeding 500 km

The `atmoshwm` function now limits the `altitude` value to 500 km when the `action` value is set to `'Warning'` or `'None'`. In previous releases, the `atmoshwm` function used `altitude` values that exceeded 500 km when `action` value was set to `'Warning'` or `'None'`.

Improved performance for computing satellite orbits

The `satelliteScenario` object shows improved simulation performance when using Two-Body-Keplerian, SGP4, or SDP4 orbit propagators. The performance improvement is observable when the scenario involves large satellite constellations involving 40 or more satellites and/or when the scenario contains more than 1000 time samples between start and stop times.

For example, compared to the previous release, the following code demonstrates that the computation of the position history of 1000 satellites using Two-Body-Keplerian, SGP4, and SDP4 orbit propagators is faster by about 51x, 64x, and 50x respectively.

```
% Create 3 satellite scenario objects. The specified start, stop, and
% sample times result in 1441 time samples in the scenario. Each scenario
% tests a specific orbit propagator.
startTime = datetime(2021,7,8);
stopTime = startTime + days(1);
sampleTime = 60;
scTBK = satelliteScenario(startTime,stopTime,sampleTime);
scSGP4 = satelliteScenario(startTime,stopTime,sampleTime);
scSDP4 = satelliteScenario(startTime,stopTime,sampleTime);

% Use the Keplerian elements to add 1000 satellites to each scenario
% with the orbit propagator in order: Two-Body-Keplerian, SGP4, and SDP4.
semiMajorAxis = ones(1,1000)*10000000;
eccentricity = ones(1,1000)*0.1;
inclination = ones(1,1000)*60;
rightAscensionOfAscendingNode = repmat(0:18:360-18,1,50);
argumentOfPeriapsis = zeros(1,1000);
trueAnomaly = sort(repmat(0:7.2:360-7.2,1,20));
satTBK = satellite(scTBK, ...
    semiMajorAxis, ...
    eccentricity, ...
    inclination, ...
    rightAscensionOfAscendingNode, ...
    argumentOfPeriapsis, ...
    trueAnomaly, ...
    "OrbitPropagator","two-body-keplerian");
satSGP4 = satellite(scSGP4, ...
    semiMajorAxis, ...
    eccentricity, ...
    inclination, ...
    rightAscensionOfAscendingNode, ...
```

```
    argumentOfPeriapsis, ...
    trueAnomaly, ...
    "OrbitPropagator","sgp4");
satSDP4 = satellite(scSDP4, ...
    semiMajorAxis, ...
    eccentricity, ...
    inclination, ...
    rightAscensionOfAscendingNode, ...
    argumentOfPeriapsis, ...
    trueAnomaly, ...
    "OrbitPropagator","sdp4");

% Initialize an array to store the satellite position in the GCRF frame.
numTimeSamples = ceil(seconds(stopTime - startTime)/sampleTime) + 1;
positionsTBK = zeros(3,numTimeSamples,1000);
positionsSGP4 = zeros(3,numTimeSamples,1000);
positionsSDP4 = zeros(3,numTimeSamples,1000);

% Time the simulation with Two-Body-Keplerian.
tic

% Fill out the position array using the states function. The first call to
% states will simulate the scenario.
for idx = 1:1000
    positionsTBK(:,:,idx) = states(satTBK(idx));
end

% Store the execution time.
executionTime = toc;

% Display the execution time.
disp("Time to compute the satellite states using Two-Body-Keplerian = " ...
                                        + executionTime + " seconds.");

% Time the simulation with SGP4.
tic

% Fill out the position array using the states function. The first call to
% states will simulate the scenario.
for idx = 1:1000
    positionsSGP4(:,:,idx) = states(satSGP4(idx));
end

% Store the execution time.
executionTime = toc;

% Display the execution time.
disp("Time to compute the satellite states using SGP4 = " ...
                              + executionTime + " seconds.");

% Time the simulation with SDP4.
tic

% Fill out the position array using the states function. The first call to
% states will simulate the scenario.
for idx = 1:1000
    positionsSDP4(:,:,idx) = states(satSDP4(idx));
end

% Store the execution time.
executionTime = toc;

% Display the execution time.
disp("Time to compute the satellite states using SDP4 = " ...
                                + executionTime + " seconds.");
```

The approximate execution times are:

R2021a: Two-Body-Keplerian – 393.49 s, SGP4 – 515.55 s, and SDP4 – 734.12 s

R2021b: Two-Body-Keplerian – 7.63 s, SGP4 – 7.95 s, and SDP4 – 14.53 s

The code was timed on a Windows® 10, Intel® Xeon® W-2133 CPU @ 3.60 GHz test system by running this script. The performance improvement can be observed in all functions related to satellite scenario objects.

Improved performance of Access object functions

The `Access` object functions (`accessPercentage`, `accessIntervals`, and `accessStatus`) show improved performance, which is observable when the number of access analysis objects in the scenario is 40 or more, and/or when the scenario contains more than 1000 time samples between start and stop times.

For example, compared to the previous release, the following code is about 29x faster.

```
% Create a satellite scenario object.
startTime = datetime(2021,7,8);
stopTime = startTime + days(1);
sampleTime = 60;
sc = satelliteScenario(startTime,stopTime,sampleTime);

% Add 40 satellites.
sat = satellite(sc,'leoSatelliteConstellation.tle');

% Point the satellite at a specified geographical coordinate.
for idx = 1:numel(sat)
    pointAt(sat(idx),[0;0;0]);
end

% Add gimbals to the satellite, and add a conical sensor to the gimbal.
for idx = 1:numel(sat)
    gim = gimbal(sat(idx));
    conicalSensor(gim);
end

% Add a ground station.
gs = groundStation(sc);

% Point the gimbals at the ground station.
gim = [sat.Gimbals];
for idx = 1:numel(gim)
    pointAt(gim(idx),gs);
end

% Add access analysis between each conical sensor and the ground station.
sensor = [gim.ConicalSensors];
for idx = 1:numel(sensor)
    access(sensor(idx),gs);
end

% Retrieve the access objects.
ac = [sensor.Accesses];

% Time the computation of the access intervals of all access objects.
tic;
intvls = accessIntervals(ac);
executionTime = toc;
disp("Execution of accessIntervals was " + executionTime + " seconds.");
```

The approximate execution times are:

R2021a: 48.18 s

R2021b: 1.64 s

The code was timed on a Windows 10, Intel Xeon W-2133 CPU @ 3.60 GHz test system by running this script.

Improved performance for satellite scenario viewer

The graphic updates on the `satelliteScenarioViewer` are observed to be faster when the satellite scenario contains 50 or more satellites.

For example, compared to the previous release, the following code for updating the visualization of 1000 satellites on the satellite scenario viewer is about 5x faster.

```
% Create a satellite scenario object.
startTime = datetime(2021,7,8);
stopTime = startTime + days(1);
sampleTime = 60;
sc = satelliteScenario(startTime,stopTime,sampleTime);

% Add 1000 satellites to the scenario using Keplerian elements.
semiMajorAxis = ones(1,1000)*10000000;
eccentricity = ones(1,1000)*0.1;
inclination = ones(1,1000)*60;
rightAscensionOfAscendingNode = repmat(0:18:360-18,1,50);
argumentOfPeriapsis = zeros(1,1000);
trueAnomaly = sort(repmat(0:7.2:360-7.2,1,20));
sat = satellite(sc, ...
    semiMajorAxis, ...
    eccentricity, ...
    inclination, ...
    rightAscensionOfAscendingNode, ...
    argumentOfPeriapsis, ...
    trueAnomaly);

% Before launching the satellite scenario viewer, hide the satellite
% labels.
for idx = 1:numel(sat)
    sat(idx).ShowLabel = false;
end

% Launch the satellite scenario viewer and hide the satellites.
v = satelliteScenarioViewer(sc);
hide(sat);

% Show the satellites and time, again.
tic;
show(sat);
updateTime = toc;
disp("The time to update the graphics was " + updateTime + " seconds.");
```

The approximate execution times are:

R2021a: 77.76 s

R2021b: 14.15 s

The code was timed on a Windows 10, Intel Xeon W-2133 CPU @ 3.60 GHz test system by running the above script.

Improved performance for field of view visualization

The visualization using `fieldOfView` function of the `ConicalSensor` object is observed to be faster. The performance improvement is observable even if there is only one field of view contour displayed, and becomes pronounced as you add more contours and/or the number of time samples between the satellite scenario start and stop times is greater than 1000.

For example, compared to the previous release, the following code for visualizing 40 field of view contours with 86401 sample times is faster by about 2.7x.

```
% Create a satellite scenario object.
startTime = datetime(2021,7,8);
```

```
stopTime = startTime + days(1);
sampleTime = 60;
sc = satelliteScenario(startTime,stopTime,sampleTime);

% Add 40 satellites.
sat = satellite(sc,"leoSatelliteConstellation.tle");

% Add conical sensors to each satellite.
for idx = 1:numel(sat)
    conicalSensor(sat(idx));
end

% Point the satellites at a specified geographical location.
for idx = 1:numel(sat)
    pointAt(sat(idx),[0;0;0]);
end

% Add field of view visualization to the conical sensors.
sensors = [sat.ConicalSensors];
fieldOfView(sensors);

% Launch the satellite scenario viewer.
v = satelliteScenarioViewer(sc);

% Pay the scenarioto visialize the evolution field of view contours, and
% time it.
tic;
play(sc);
executionTime = toc;
disp("The time taken to calculate and visualize the field of view contours is " ...
                                                    + executionTime + " seconds.");
```

The approximate execution times are:

R2021a: 102.52 s

R2021b: 37.56 s

The code was timed on a Windows 10, Intel Xeon W-2133 CPU @ 3.60 GHz test system by running this script.

Functionality being removed or changed

atmoshwm function possible changed returned values

Behavior change

The `atmoshwm` function now:

- Accepts `day` decimal input values.
- Limits `altitude` input values to `500` km.

As a result, the output values from this function might change from previous releases.

atmoscoesa function changed input and returned value formats

Behavior change

The `atmoscoesa` function now:

- Accepts scalar, vector, or matrix values.
- Outputs scalar, vector, or matrix values.

As a result, the output values from this function might change from previous releases.

R2021a

New Aero.FixedWing class to define fixed-wing aircraft

The new `Aero.FixedWing` class defines fixed-wing aircraft. Use this class in conjunction with supporting classes `Aero.FixedWing.Coefficient`, `Aero.FixedWing.State`, `Aero.FixedWing.Surface`, `Aero.FixedWing.Thrust`, `Aero.Aircraft.Properties`, `Aero.Aircraft.Environment`, and `Aero.Aircraft.ControlState` to:

- Define aircraft dynamics.
- Define aircraft dynamics from DATCOM files.
- Perform static stability analyses.
- Generate state-space representation with linearization methods.
- Integrate more easily with Control System Toolbox™ workflows.

New object to create satellite scenario objects

Use the new satelliteScenario object to:

- Define satellites and their orbits.
- Define ground stations.
- Visualize satellites in orbit and ground tracks.
- Visualize satellite field-of-view on Earth.
- Analyze line-of-sight access between satellites and ground stations.
- View satellite scenario with playback animation.

For more information, see Satellite Scenario Overview.

Functions to support spacecraft applications

New functions support spacecraft applications in conjunction with Aerospace Blockset™:

- `ecef2eci`
- `eci2ecef`
- `ijk2keplerian`
- `keplerian2ijk`
- `siderealTime`

These functions were previously part of the CubeSat Simulation Library Add-On.

R2020b

Version: 3.4

New Features

Bug Fixes

Version History

Updated igrfmagm function

The `igrfmagm` function has been updated to support the International Geomagnetic Reference Field 13 (IGRF-13) model. For more information on the changes, see “igrfmagm function behavior changes” on page 4-2.

FlightGear support updates

The Aerospace Toolbox FlightGear object no longer requires the specification of particular FlightGear versions. Aerospace Toolbox supports FlightGear versions starting at V2.6. As a result, the `FlightGearVersion` property has been removed from the `Aero.FlightGearAnimation Objects` object and `GenerateRunScript (Aero.FlightGearAnimation)` method.

geoc2geod and geod2geoc functions have new output arguments

The `geoc2geod` function has a new output argument that enables the output of the mean sea-level altitude (MSL).

The `geod2geoc` function has a new output argument that enables the output of the radius from the center of the planet to the center of gravity.

geoidheight function change

Starting in R2020b, use a geodetic latitude for the `geoidheight latitude` argument. In previous releases, you were directed to use a geocentric latitude. Using a geodetic latitude might output different results.

Functionality being removed or changed

igrfmagm function behavior changes

Behavior change

The `igrfmagm` function has been updated to support the IGRF-13 model, which introduces these changes:

- Results of the function might differ from previous releases.
- The function now accepts matrices as inputs. In previous releases, the function accepted only scalar values.
- The function allows higher height value to 5.6 Earth radii (35,717,567.2 m). Previously, this value was 600,000 m.
- The function allows a wider range of latitude values (greater than 90, less than -90). When approaching the poles, the function generates more accurate data than in previous releases. The `igrfmagm` function no longer generates NaNs when input values approach the poles.
- The function allows wider range of longitude values (greater than 180, less than -180).

Updated aeroiersdata.mat file

Behavior change

The contents of the `aeroiersdata.mat` file have been updated. Correspondingly, the output of the `deltaUT1`, `deltaCIP`, and `polarMotion` functions will have different results when using the default

value (`'aeroiersdata.mat'`) as the value of `Source`. The results reflect more accurate external data from the International Earth Rotation and Reference Systems Service (IERS).

R2020a

wrldmagm function support for World Magnetic Model 2020

The `wrldmagm` function now supports World Magnetic Model 2020 by default.

FlightGear interface supports Version 2019.1

The Aerospace Toolbox product now supports FlightGear v2019.1.

geoc2geod and geod2geoc function updates

The `geoc2geod` and `geod2geoc` functions no longer use a low altitude approximation.

Version History

The `geoc2geod` and `geod2geoc` functions no longer use a low altitude approximation. This change results in geodetic latitude (`geoc2geod`) and geocentric latitude (`geod2geoc`) output being more accurate at higher altitudes than in previous releases.

Aerospace Toolbox Flight Instrument Gauges New Property

The flight instrument gauge properties support a new property, `uicontextmenu`, which adds and configures context menu components in apps and on the App Designer canvas. For more information, see “uicontextmenu Function: Add and configure context menu components in apps and on the App Designer canvas”.

R2019b

Version: 3.2

New Features

Bug Fixes

wrldmagm function support for World Magnetic Model 2015v2

The `wrldmagm` function now supports World Magnetic Model 2015v2 by default. In addition, you can use the new argument, `'Custom'`, to directly specify a coefficient file provided by NOAA to the function.

WMM2015v2 supersedes WMM2015(v1). Consider replacing WMM2015(v1) with WMM2015v2 when used for navigation and other systems. WMM2015v2 was released by NOAA in February, 2019 to correct performance degradation issues in the Arctic region for January 1, 2015 to December 31, 2019. Therefore, it is still acceptable to use WMM2015(v1) in systems below 55-degrees latitude in the Northern hemisphere.

Existing applications have this behavior:

- If the `wrldmagm` function has the `model` argument set to `'2010'`, `'2005'`, or `'2000'`, the application continues to work as before.
- If the `wrldmagm` has the `model` argument set to `'2015'` or default, the application uses the WMM2015v2 coefficient file.

FlightGear interface supports Version 2018.3

The Aerospace Toolbox product now supports FlightGear v2018.3.

If you do not download scenery in advance, you can direct FlightGear to download it automatically during simulation using the `InstallScenery` property of the `Aero.FlightGearAnimation` object for the `GenerateRunScript (Aero.FlightGearAnimation)` method.

Starting with FlightGear v2018.3 on Windows systems, you may encounter an error message while launching FlightGear with the `InstallScenery` option enabled:

```
Error creating directory: No such file or directory
```

This error likely indicates that your default FlightGear download folder is not writeable, the path cannot be resolved, or the path contains UNC path names. To work around the issue, edit the `runfg.bat` file to specify a new folder path to store the scenery data:

1. Edit `runfg.bat`.
2. To the list of command options, append `--download-dir=` and specify a folder to which to download the scenery data. For example:

   ```
   --download-dir=C:\Users\user1\Documents\FlightGear
   ```

   All data downloaded during this FlightGear session is saved to the specified directory. To avoid downloading duplicate scenery data, use the same directory in succeeding FlightGear sessions
3. To open FlightGear, run `runfg.bat`.

Note Each time that you run the `GenerateRunScript` function, it creates a new script. It overwrites any edits that you have added.

R2019a

Version: 3.1

New Features

Bug Fixes

Aerospace Toolbox flight instrument gauges available in App Designer

Aerospace Toolbox flight instrument gauges are now available in App Designer in the component library. Creation of applications using these flight instrument gauges requires an Aerospace Toolbox license. For more information, see Flight Instrument Components in App Designer and App Designer (MATLAB).

FlightGear interface supports Version 2018.2

The Aerospace Toolbox product now supports FlightGear v2018.2.

R2018b

Flight Instruments: Display measurements in UI figure windows using standard cockpit instruments

Use these functions, representing standard cockpit instruments,and their associated property pages, to display measurements:

Function	Property Page
`uiaeroairspeed`	`AirspeedIndicator Properties`
`uiaeroaltimeter`	`Altimeter Properties`
`uiaeroclimb`	`ClimbIndicator Properties`
`uiaeroegt`	`EGTIndicator Properties`
`uiaeroheading`	`HeadingIndicator Properties`
`uiaerohorizon`	`ArtificialHorizon Properties`
`uiaerorpm`	`RPMIndicator Properties`
`uiaeroturn`	`TurnCoordinator Properties`

For an example of these components, see the `Display Flight Trajectory Data Using Flight Instruments and Flight Animation` example.

For more information, see Flight Instruments.

Polar Motion: Calculate the movement of rotation axis with respect to the Earth crust according to IAU2000A

Use `polarMotion` to calculate the movement of the rotation axis with respect to the crust of the Earth for a specific Universal Coordinated Time (UTC), according to the IAU2000A reference system.

Supersonic Airspeed Correction: Convert between equivalent, calibrated, or true airspeed

The `correctairspeed` function has been updated to now also work with supersonic airspeeds. The function also now lets you choose a method for computing the conversion factor (table lookup or compute on demand).

Version History

The `correctairspeed` function output may differ from the previous version of the function.

For a potentially more accurate output, consider using the equation method.

Celestial Intermediate Pole Location: Calculate adjustment to the celestial intermediate pole location according to IAU2000A

Use `deltaCIP` to calculate the adjustment to the celestial intermediate pole location according for a specific Universal Coordinated Time (UTC), according to the IAU2000A reference system.

FlightGear Interface: Includes support for Version 2018.1 through flight simulator objects

The Aerospace Toolbox product now supports FlightGear v2018.1.

R2018a

FlightGear Interface: Includes support for Version 2017.3 through flight simulator objects

The Aerospace Toolbox product now supports FlightGear v2017.3.

Animation objects changes

Changes in the use of Aerospace Toolbox animation objects:

- Aerospace Toolbox animation objects saved in MAT-files in R2018a cannot load in previous releases.
- The `delete` function for all Aerospace Toolbox animation objects now destroys the animation object. In previous releases, the object was not destroyed.

Version History

Scripts or functions creating Aerospace Toolbox animation objects continue to work. Use these scripts or functions to create new objects.

Function or File	What Happens When You Use the Function or File?	Use These Functions or Files Instead	Compatibility Considerations
MAT-files that contain animation objects	Warns	Scripts or functions creating new animation objects	Animation objects saved in MAT-files in R2018a will not load in previous releases.
`delete`	Object is destroyed	None	The `delete` function for all animation objects now destroys the object. In previous releases, the object was not destroyed.

Direction cosine matrix validity checks

These functions can now verify the validity of the direction cosine matrix prior to conversion:

- `dcm2alphabeta`
- `dcm2angle`
- `dcm2latlon`
- `dcm2quat`
- `dcm2rod`

Each function now lets you specify the error tolerance level for the direction cosine matrix validation and specify an action if the matrix is not valid.

Install FlightGear scenery during simulation

When you install the FlightGear software, the installation provides a basic level of scenery files. The FlightGear documentation guides you through installing scenery as part the general FlightGear

installation. If you do not download scenery, you can direct FlightGear to download it automatically during simulation using the `InstallScenery` property of the `Aero.FlightGearAnimation` object for the `GenerateRunScript (Aero.FlightGearAnimation)` method.

Disable FlightGear shaders

Your computer built-in video card, such as NVIDIA® cards, can conflict with FlightGear shaders. You can disable the FlightGear shaders by specifying the `DisableShaders` property of the `Aero.FlightGearAnimation` object to the `GenerateRunScript (Aero.FlightGearAnimation)` method.

R2017b

Version: 2.20

New Features

Bug Fixes

Difference Between UT1 and UTC: Calculate time difference with deltaUT1 function according to the IAU2000A reference system

Use `deltaUT1` to calculate the difference between principal Universal Time (UT1) and Coordinated Universal Time (UTC) according to the IAU2000A reference system.

To optionally create a file containing the current Earth orientation data for `deltaUT1`, use the `aeroReadIERSData` function.

FlightGear Interface: Includes support for Version 2017.1 through flight simulator objects

The Aerospace Toolbox product now supports FlightGear v2017.1.

For more information on working with FlightGear, see Aero.FlightGearAnimation Objects.

quat2angle and rod2angle updates

The `quat2angle` and `rod2angle` functions now return values for the middle angle of the 'ZYZ', 'ZXZ', 'YXY', 'YZY', 'XYX', and 'XZX' implementations. In previous releases, these functions returned all zeroes for 0 degrees in the second rotation.

R2017a

Euler-Rodrigues Functions: Convert to and from Rodrigues vectors

These functions convert Euler-Rodrigues vectors to and from direction cosine matrices, rotation angles, and quaternions:

- `angle2rod`
- `dcm2rod`
- `quat2rod`
- `rod2angle`
- `rod2dcm`
- `rod2quat`

FlightGear Interface: Includes support for Version 2016.3 through flight simulator objects

The Aerospace Toolbox product now supports FlightGear v2016.3.

For more information on working with FlightGear, see Aero.FlightGearAnimation Objects.

Aerospace Toolbox Software and MATLAB string

The Aerospace Toolbox software now supports MATLAB `string`.

angle2quat Function Replaces euler2quat

The `angle2quat` function has replaced the `euler2quat` function.

Version History

The `euler2quat` function is no longer available. Use the `angle2quat` function instead.

R2016b

Horizontal Wind Model 14 Function: Calculate meridional and zonal wind components using U.S. Naval Research Laboratory HWM14 model

The `atmoshwm` function implements horizontal wind modes.

Version History

The `atmoshwm` replaces the `atmoshwm07` function.

Function or Function Element Name	What Happens When You Use the Function or Element?	Use These Functions or Function Elements Instead	Compatibility Considerations
`atmoshwm07`	Warns	`atmoshwm`	To use a specific generation of the Horizontal Wind Model, specify the appropriate year in the `atmoshwm` function.

FlightGear Version 2016.1 Support: Interface with FlightGear through flight simulator object

The Aerospace Toolbox product now supports FlightGear v2016.1.

For more information on working with FlightGear, see Aero.FlightGearAnimation Objects.

R2016a

Quaternion Interpolation: Calculate interpolation between two quaternions

The `quatinterp` function interpolates between two quaternions. To support this function, the following functions are also new:

The `quatpower` function calculates the power of a quaternion.

The `quatlog` function calculates the natural logarithm of a quaternion.

The `quatexp` function calculates the exponential of a quaternion.

FlightGear versions earlier than 2.0 no longer supported

The Aerospace Toolbox software no longer supports FlightGear versions earlier than 2.0. For a list of FlightGear versions that the Aerospace Toolbox software supports, see Supported FlightGear Versions.

Version History

If you are using a FlightGear version older than 2.0, update your FlightGear installation to a supported version. The software returns an error if you use a non-supported version. Obtain updated FlightGear software from `www.flightgear.org` in the download area.

Unit conversion function precision changes

These unit conversion functions may now generate values with better precision:

- `convacc`
- `convang`
- `convangacc`
- `convangvel`
- `convdensity`
- `convforce`
- `convlength`
- `convmass`
- `convpres`
- `convtemp`
- `convvel`

R2015b

FlightGear Versions 3.2 and 3.4 Support: Interface to these FlightGear versions through flight simulator object

The Aerospace Toolbox product now supports FlightGear v3.2 and v3.4.

For more information on working with FlightGear, see Aero.FlightGearAnimation Objects.

igrfmagm Function: Calculate Earth magnetic field and secular variation

The `igrfmagm` function calculates Earth magnetic field and secular variation using the International Geomagnetic Reference Field.

Version History

The `igrfmagm` replaces the `igrf11magm` function. For more information, see “Function and Function Element Being Removed” on page 14-2.

DATCOM 2014 Support: Import aerodynamic coefficients from this version

The `datcomimport` function has been enhanced to support the 2014 version of DATCOM files.

Function and Function Element Being Removed

The following table lists the function being removed for R2015b.

Function or Function Element Name	What Happens When You Use the Function or Element?	Use These Functions or Function Elements Instead	Compatibility Considerations
`igrf11magm`	Warns	`igrfmagm`	To use a specific generation of the International Geomagnetic Reference Field, specify the appropriate year in the `igrfmagm` function.

R2015a

Version: 2.15

New Features

Bug Fixes

tdbjuliandate function that calculates barycentric dynamical time for a given terrestrial time date

The `tdbjuliandate` function calculates Barycentric Dynamical Time (TDB) for a given Terrestrial Time (TT) date.

eci2aer function that converts Earth-centered inertial coordinates to azimuth, elevation, and range

The `eci2aer` function converts Earth-centered inertial coordinates to azimuth, elevation, and range.

Additional ephemerides coefficient support for celestial phenomena functions

The `moonLibration`, and `planetEphemeris` functions now support the DE432t ephemerides database.

World Magnetic Model 2015 support

The `wrldmagm` function supports the world magnetic model for 2015 to 2020 (WMM-2015).

New example

The `Estimate Sun Analemma Using Planetary Ephemerides and ECI to AER Transformation` example shows how to estimate the Sun analemma using the `eci2aer` and `tdbjuliandate` functions.

R2014b

Version: 2.14

New Features

Bug Fixes

atmoshwm07 function for Horizontal Wind Model 07 data

The `atmoshwm07` function implements the U.S. Naval Research Laboratory HWM™ routine to calculate the meridional and zonal components of the wind for a set of geophysical data.

FlightGear animation object support for FlightGear Version 3

The Aerospace Toolbox product now supports FlightGear Version 3.

For more information on working with FlightGear, see Aero.FlightGearAnimation Objects.

Additional ephemerides coefficient support for celestial phenomena functions

The `planetEphemeris`, `earthNutation`, and `moonLibration` functions now support the DE430 ephemerides coefficient.

R2014a

Version: 2.13

New Features

Bug Fixes

Functions to convert between latitude, longitude, altitude and Earth-centered inertial coordinates

The `lla2eci` function converts geodetic latitude, longitude, altitude (LLA) coordinates to Earth-centered inertial (ECI) position coordinates, based on the specified reduction method and Universal Coordinated Time (UTC), for the specified time and geophysical data.

The `eci2lla` function converts Earth-centered inertial (ECI) position coordinates to geodetic latitude, longitude, altitude (LLA) coordinates, based on the specified reduction method and Universal Coordinated Time (UTC), for the specified time and geophysical data.

FlightGear animation object support for FlightGear Version 2.12

The Aerospace Toolbox product now supports FlightGear Version 2.12.

For more information on working with FlightGear, see Aero.FlightGearAnimation Objects.

Add ephemeris and geoid data

Use the `aeroDataPackage` function to add ephemeris and/or geoid data for these Aerospace Toolbox functions and Aerospace Blockset blocks.

Aerospace Toolbox Functions	Aerospace Blockset Blocks
`geoidheight` **Note** Only for the EGM2008 Geopotential Model. Aerospace Toolbox provides EGM96 Geopotential Model data.	Geoid Height **Note** Only for the EGM2008 Geopotential Model. Aerospace Toolbox provides EGM96 Geopotential Model data.
`earthNutation`	Earth Nutation
`moonLibration`	Moon Libration
`planetEphemeris`	Planetary Ephemeris

R2013b

FlightGear animation object support for FlightGear Version 2.10

The Aerospace Toolbox product now supports FlightGear Version 2.10.

For more information on working with FlightGear, see Aero.FlightGearAnimation Objects.

dcmeci2ecef function to convert Earth-Centered Inertial to Earth-Centered Earth-Fixed coordinates

The `dcmeci2ecef` function calculates the position direction cosine matrix (ECI to ECEF), based on the specified reduction method and Universal Coordinated Time (UTC), for the specified time and geophysical data.

Latitude inputs outside +90 and -90 degrees

These functions now correctly take into account latitude inputs that are outside +90 and -90 degrees.

- `atmosnrlmsise00`
- `geoc2geod`
- `geod2geoc`
- `flat2lla`
- `lla2flat`
- `geoidheight`
- `gravitywgs84`

Celestial navigation example

The Marine Navigation Using Planetary Ephemerides example shows how to perform celestial navigation of a marine vessel using the planetary ephemerides and Earth-Centered Inertial to Earth-Centered Earth-Fixed (ECI to ECEF) transformation.

Phaseout of FlightGear versions earlier than 2.0

The Aerospace Toolbox software will not support FlightGear versions earlier than 2.0 in a future release of Aerospace Toolbox. For a list of FlightGear versions that the Aerospace Toolbox software supports, see Supported FlightGear Versions.

Version History

If you are using a FlightGear version older than 2.0, update your FlightGear installation to a supported version.

R2013a

Version: 2.11

New Features

Bug Fixes

FlightGear animation object support for FlightGear Version 2.8

For more information on working with FlightGear, see `Aero.FlightGearAnimation`.

planetEphemeris function to implement position and velocity of Solar System planets

The `planetEphemeris` function implements the position and velocity of an astronomical object.

earthNutation function to implement nutation in longitude and obliquity of Earth

The `earthNutation` function implements the nutation in longitude and obliquity of Earth according to the International Astronomical Union (IAU) 1980 nutation series.

moonLibration function to implement relative motion attitude of Moon

The `moonLibration` function implements the relative motion attitude of Earth's Moon.

Recorded video of Aero.Animation or Aero.VirtualRealityAnimation objects for playback later

You can now record flight data animations for `Aero.Animation` and `Aero.VirtualRealityAnimation` objects. The following properties are new for both classes:

Property	Description
`VideoRecord`	Enable video recording.
`VideoFileName`	Specify video recording file name.
`VideoCompression`	Specify video recording compression file type.
`VideoQuality`	Specify video recording quality.
`VideoTStart`	Specify video recording start time for scheduled recording.
`VideoTFinal`	Specify video recording stop time.

Architecture selection support for Aero.FlightGearAnimation

You can specify the architecture the FlightGear software is running on. `GenerateRunScript` takes this setting into account when generating the run script. These architecture settings are available.

Architecture	Setting
Windows (32-bit)	`'Win32'`
Windows (64-bit) architecture.	`'Win64'`
Mac OS X (64-bit) architecture.	`'Mac'`
Linux® (64-bit) architecture.	`'Linux'`

R2012b

Version: 2.10

New Features

Bug Fixes

FlightGear animation object support for FlightGear versions 2.4 and 2.6

The Aerospace Toolbox product now supports FlightGear Versions 2.6 and 2.4.

For more information on working with FlightGear, see Aero.FlightGearAnimation Objects.

R2012a

Version: 2.9

New Features

Support 2011 Version of DATCOM

The `datcomimport` function has been enhanced to support the 2011 version of DATCOM files.

Using FlightGear Version 2.4.0 with Aerospace Toolbox

Aerospace Toolbox Version 2.9 does not support FlightGear Version 2.4.0. Use this procedure as a workaround.

1. In the MATLAB Command Window, create a FlightGear animation object.

   ```
   h = Aero.FlightGearAnimation;
   ```

2. Set the FlightGear animation object property `FlightGearVersion` to 2.0.

   ```
   h.FlightGearVersion = '2.0';
   ```

3. Set the FlightGear animation object property `FlightGearBaseDirectory` to the location of FlightGear Version 2.4.0.

   ```
   h.FlightGearBaseDirectory = 'C:\Program Files\FlightGear240'
   ```

4. Generate the run script.

   ```
   GenerateRunScript(h)
   ```

5. Save and close this file.

For more information, see Aero.FlightGearAnimation Objects in the Aerospace Toolbox User's Guide.

R2011b

Version: 2.8

New Features

Version History

Conversion of Error and Warning Message Identifiers

For R2011b, error and warning message identifiers have changed in Aerospace Toolbox.

Version History

If you have scripts or functions that use message identifiers that changed, you must update the code to use the new identifiers. Typically, message identifiers are used to turn off specific warning messages.

For example, the `Aero:FlightGearAnimation:NeedTimeData` identifier has changed to `aero:FlightGearAnimation:NeedTimeData`. If your code checks for `Aero:FlightGearAnimation:NeedTimeData`, you must update it to check for `aero:FlightGearAnimation:NeedTimeData` instead.

To determine the identifier for a warning, run the following command just after you see the warning:

```
[MSG,MSGID] = lastwarn;
```

This command saves the message identifier to the variable *MSGID*.

Note Warning messages indicate a potential issue with your code. While you can turn off a warning, a suggested alternative is to change your code so it runs warning-free.

Demos

The following demos are new:

- `Visualizing World Magnetic Model Contours for 2010 Epoch` — Visualize contour plots of the calculated values for the Earth's magnetic field using World Magnetic Model 2010 (WMM-2010) overlaid on maps of the Earth.
- `Visualizing Geoid Height for Earth Geopotential Model 1996` — Calculate the Earth Geoid height using the EGM96 Geopotential Model.

Function and Function Element Being Removed

The following table lists the function and function element name being removed for R2011b.

Function or Function Element Name	What Happens When You Use the Function or Element?	Use These Functions or Function Elements Instead	Compatibility Considerations
`wrldmagm '2000'` or `'2005'` epoch year	Warns	For model years between 2000 and the start of 2010, use `igrf11magm`. For model years between 2010 and the start of 2015, use `wrldmagm`.	For model years between 2000 and the start of 2010, use `igrf11magm`. For model years between 2010 and the start of 2015, use `wrldmagm`.

R2011a

Version: 2.7

New Features

New LLA to Flat Earth Function

The `lla2flat` function estimates a flat Earth position from geodetic latitude, longitude, and altitude coordinates.

New Flat Earth to LLA Function

The `flat2lla` function estimates geodetic latitude, longitude, and altitude coordinates from a flat Earth position.

New International Geomagnetic Reference Field 11 Function

The `igrf11magm` function calculates the Earth's magnetic field using the 11th generation of the International Geomagnetic Reference Field.

The gravityspericalharmonic Function Supports New Planet Model

The `gravityspericalharmonic` function now supports the EIGEN-GL04C gravity field model.

R2010b

Version: 2.6

New Features

Version History

New Geoid Height Function

The `geoidheight` function calculates the height of geoid undulations/height using one of three geopotential models.

Support to Read File Types 6, 21, and 42 for 2008 Version of DATCOM

The `datcomimport` function has been enhanced to read file types 6, 21, and 42 for 2008 DATCOM files. In previous releases, the Aerospace Toolbox read only file type 6 and 21.

Support for FlightGear 2.0

Aerospace Toolbox now supports FlightGear Version 2.0.

For more information on working with FlightGear, see Aero.FlightGearAnimation Objects in the Aerospace Toolbox User's Guide.

Functions and Function Elements Being Removed

Function or Function Element Name	What Happens When You use the Function or Element?	Use This Instead	Compatibility Considerations
`geoidegm96`	Warns	`geoidheight`	Replace all existing instances of `geoidegm96` with `geoidheight`.

R2010a

Version: 2.5

New Features

New Gravity Centrifugal Effect Function

The `gravitycentrifugal` function implements the centrifugal effect for eight planets and the Moon, plus the capability to customize this effect.

New Spherical Harmonic Gravity Model Function

The `gravitysphericalharmonic` function implements the spherical harmonic gravity models for Earth (EGM2008, EGM96), Moon (LP100K, LP165P), and Mars (GMM2B), plus the capability to customize these models.

New Gas Dynamics Functions

New gas dynamics functions, including isentropic flow (`flowisentropic`), normal shock (`flownormalshock`), Rayleigh flow (`flowrayleigh`), Fanno flow (`flowfanno`), and Prandtl-Meyer flow (`flowprandtlmeyer`).

Updated World Magnetic Function

Updated `wrldmagm` function to include world magnetic model for years 2010-2015 (WMM-2010).

Demos

The `Comparing Zonal Harmonic Gravity Model to Other Gravity Models` demo has been updated to include comparison of other gravity models.

R2009b

Version: 2.4

New Features

New Zonal Harmonic Gravity Model Function

The `gravityzonal` function implements the zonal harmonic gravity model.

Support for FlightGear 1.9.1

Aerospace Toolbox Version 3.4 now supports FlightGear Version 1.9.1.

For more information on working with FlightGear, see Aero.FlightGearAnimation Objects in the Aerospace Toolbox User's Guide.

R2009a

Version: 2.3

New Features

Support to Read File Type 21 for 2007 Version of DATCOM

The `datcomimport` function has been enhanced to read file type 21 for 2007 DATCOM files. In previous releases, the Aerospace Toolbox read only file type 6.

Using FlightGear Version 1.9.0 with Aerospace Toolbox

Aerospace Toolbox Version 2.3 does not support FlightGear Version 1.9.0. You can use this procedure.

1. In the MATLAB Command Window, create a FlightGear animation object.

   ```
   h = Aero.FlightGearAnimation;
   ```

2. Set the FlightGear animation object property `FlightGearVersion` to 1.0.

   ```
   h.FlightGearVersion = '1.0';
   ```

3. Set the FlightGear animation object property `FlightGearBaseDirectory` to the location of FlightGear Version 1.9.0.

   ```
   h.FlightGearBaseDirectory = 'C:\Program Files\FlightGear190'
   ```

4. Generate the run script.

   ```
   GenerateRunScript(h)
   ```

5. Open the custom FlightGear run script with a text editor and change the input parameter `'--airport-id='` to `'--airport='`.
6. Save and close this file.

For more information on working with FlightGear, see Aero.FlightGearAnimation Objects in the Aerospace Toolbox User's Guide.

R2008b

Version: 2.2

New Features

Support for 2007 Version of DATCOM File

The `datcomimport` function has been enhanced to support the 2007 DATCOM file in addition to the 1976 and 1999 DATCOM files.

FlightGear Version 1.0 with Aerospace Toolbox

Aerospace Toolbox Version 2.2 now supports FlightGear Version 1.0. To access this version of FlightGear, you can use this procedure.

1. In the MATLAB Command Window, create a FlightGear animation object.

   ```
   h = Aero.FlightGearAnimation;
   ```

2. Set the FlightGear animation object property `FlightGearVersion` to 1.0.

   ```
   h.FlightGearVersion = '1.0';
   ```

3. Set the FlightGear animation object property `FlightGearBaseDirectory` to the location of FlightGear Version 1.0.

   ```
   h.FlightGearBaseDirectory = 'C:\Program Files\FlightGear10'
   ```

For more information on working with FlightGear, see Aero.FlightGearAnimation Objects in the Aerospace Toolbox User's Guide.

FlightGear Animation Object play Method Now Supports Custom Timers

The FlightGear animation object `play` method now supports custom timers.

In previous releases, you needed to create your own `play` method if your FlightGear animation object was used with custom timers. This is no longer necessary.

R2008a

Version: 2.1

New Features

Support for 1999 Version of DATCOM File

The `datcomimport` function has been enhanced to support the 1999 DATCOM file in addition to the 1976 DATCOM file.

Using FlightGear Version 1.0 with Aerospace Toolbox

Aerospace Toolbox Version 2.1 does not support FlightGear Version 1.0. You can use this procedure.

1. In the MATLAB Command Window, create a FlightGear animation object.

   ```
   h = Aero.FlightGearAnimation;
   ```

2. Set the FlightGear animation object property `FlightGearVersion` to 0.9.10.

   ```
   h.FlightGearVersion = '0.9.10';
   ```

3. Set the FlightGear animation object property `FlightGearBaseDirectory` to the location of FlightGear Version 1.0.

   ```
   h.FlightGearBaseDirectory = 'C:\Program Files\FlightGear10'
   ```

For more information on working with FlightGear, see Aero.FlightGearAnimation Objects in the Aerospace Toolbox User's Guide.

R2007b

Version: 2.0

New Features

Version History

Virtual Reality Toolbox Animation Object

This release introduces the following new objects and their associated methods to visualize flight data using the Virtual Reality Toolbox™ product:

- `Aero.VirtualRealityAnimation`
- `Aero.Node`
- `Aero.Viewpoint`

Support for the COSPAR International Reference Atmosphere 1986 Model

The `atmoscira` function implements the COSPAR International Reference Atmosphere (CIRA) 1986 environmental model.

Support for 2001 United States Naval Research Laboratory Mass Spectrometer and Incoherent Scatter Radar Exosphere

The `atmosnrlmsise00` function implements the 2001 United States Naval Research Laboratory Mass Spectrometer and Incoherent Scatter Radar Exosphere (NRLMSISE) environmental model.

Support for the EGM96 Geopotential Model

The `geoidegm96` function implements the 1996 Earth Geopotential Model (EGM96).

quat2angle Function Replaces quat2euler

The `quat2angle` function converts spatial representation from any of 12 standard sequences of rotation angles to quaternions.

Version History

The `quat2euler` function is deprecated. Applications that contain this function continue to be supported, but an error message will be displayed. Use the `quat2angle` function instead.

angle2quat Function Replaces euler2quat

The `angle2quat` function converts spatial representation from quaternions to any of 12 standard sequences of rotation angles.

Version History

The `euler2quat` function is deprecated. Applications that contain this function continue to be supported, but an error message will be displayed. Use the `angle2quat` function instead.

R2007a

Version: 1.1

New Features

New Aerospace Toolbox Objects

This release introduces the following new objects and their associated methods to create a six–degrees–of–freedom animation of multiple bodies that have custom geometries:

- `Aero.Animation`
- `Aero.Body`
- `Aero.Camera`
- `Aero.Geometry`

New Aerospace Toolbox Demo

The Aerospace Toolbox product has a new demo, Overlaying Simulated and Actual Flight Data, which illustrates the use of the Aero objects.

R2006b

Version: 1.0

New Features

Introduction of Aerospace Toolbox Product

This product extends the MATLAB technical computing environment by providing reference standards, environment models, and aerodynamic coefficient importing for performing advanced aerospace analysis to develop and evaluate your designs. An interface to the FlightGear flight simulator enables you to visualize flight data in a three-dimensional environment and reconstruct behavioral anomalies in flight-test results. To ensure design consistency, the Aerospace Toolbox software provides utilities for unit conversions, coordinate transformations, and quaternion math, as well as standards-based environmental models for the atmosphere, gravity, and magnetic fields. You can import aerodynamic coefficients directly from the U.S. Air Force Digital Data Compendium (DATCOM) to carry out preliminary control design and vehicle performance analysis.

The toolbox provides you with the following main features:

- Provides standards-based environmental models for atmosphere, gravity, and magnetic fields.
- Converts units and transforms coordinate systems and spatial representations.
- Implements predefined utilities for aerospace parameter calculations, time calculations, and quaternion math.
- Imports aerodynamic coefficients directly from the U.S. Air Force Digital Data Compendium (DATCOM).
- Interfaces to the FlightGear flight simulator, enabling visualization of vehicle dynamics in a three-dimensional environment.

The Aerospace Toolbox software has the following limitation:

- The FlightGear animation object cannot be compiled with the MATLAB Compiler software to create a standalone application.