G-NAV

THE WEB BASED FLIGHT COMPUTER
GENERAL SPECIFICATIONS AND USER'S MANUAL

March 2025

Evaluation and demonstration stage

Version 4A

Preface

G-NAV is a progressive web application (PWA) that can be used for air navigation. The system is developed under a non-profit premise and the software is free and open source.

The ultimate goal of G-NAV is that of increasing the situational awareness of glider pilots and reducing the risks of the sport. Hopefully you can find in this application a useful companion for having fun safe flights.

Remember that a tool is only useful when you know how to use it properly, so if you are planning to use G-NAV, take the necessary time to read this manual from the top to the end and to practice sufficiently before carrying the tool on board. Make sure you know how every function works before using it.

Note that G-NAV is an ongoing endeavor. This document often describes features that are not fully complete.

Design philosophy

This project seeks to develop a simple, robust and safe application. It is not the intention to substitute the functionality of other on-board equipment, like altimeter or variometer, but rather to give easy access to a complementary tool based on a different sensor, namely the GNSS.

Plenty of features can be included in a soaring computer, however, every new function demands more code, higher risk of failures, more training and more attention during operations. Therefore, if the number of functions is too large, the product becomes hard to maintain and operate. A limited number of features, on the other hand, results in a more stable and resilient tool.

The main design philosophy in G-NAV is to avoid at maximum features that can potentially confuse or disrupt the attention of the pilot. Additionally, the functions are deliberately limited and simplified to reduce the amount of source code, since every algorithm is a potential source of bugs.

These are some of the design rules and guidelines that are being consciously followed in the development of the application. If you take a look at this, you will understand why the application is the way it is. We are conscious that the app could still be improved, but we will always move forward in this direction.

- Allowing the adaptation of the graphical interface is in general a bad idea from a safety point of view, and it is even worse if you allow pilots to do that while flying. It not only requires more source code, but it can also potentially confuse the pilot. Data must preferably be at a fixed place and always use a fixed rule for colour and representation.
  In G-NAV, changing the function or the position of the data is simply not possible unless you change the source code.
- Animations demand in general a high level of resources. Even worse, they increase the complexity of the code, while most of the time they provide little to none added value in a critical situation. For example, if you activate a function by pressing something, what you immediately want to see is the new state of the function, not if you pressed the button, which you obviously did.
  The only animation G-NAV implements is the blinking of data fields or messages. For the rest, the application is based on its own widgets library that is extremely frugal and deliberately free of these features.
- Functions must be readable and to the point. If something isn't readable or hard to interpret, it will only consume more time for the pilot to use.
- Give colors a meaning and be consequent on that. Colors are part of our perception and can be used as an extra information channel. If everything looks the same, the pilot will lose more time trying to find the data he needs.
- Implement a smooth degradation when possible. Try to provide alternatives to functions that depend on sensors or specific data. For example, if the GPS is not

reporting altitude, but it still reports the position, then you won't know how high you'll be at the next waypoint, but you will know how much altitude you'll lose. Since you have an altimeter on-board, you'll still be able to figure out the other. Or if the wind cannot be calculated, then let the user enter a manual value. There are several examples like this.

- Data that is not available should not be used, and the pilot should be aware of that. If some data is not valid, dependent data should neither be valid. The absence or invalidity of a data source must be propagated to dependent data fields.
- You don't need to inform the pilot about something he already knows. For example, poping-up a message to notify takeoff or landing is completely useless and distracting.
- Post processing of the flight data doesn't belong to an on-board flight instrument, because the pilot will not and should not do that job during flight. So this is better left for a separate ground application.
- Navigation through pages must be limited to a minimum. A flight computer isn't a desktop application. If the pilot gets lost in the pages trying to find a function, he won't be looking outside for a while. In G-NAV there are therefore only two levels of pages and there is always a red button to return immediately to the navigation.
- Implement custom widgets to minimize the number of inputs and distraction time from the pilot. Most standard graphic toolkits are not really suitable for the requirements during flight, simply because they were designed with other ideas in mind. Therefore, in G-NAV we prefer to build our own data widgets.
- Crashes or malfunctioning due to wrong input data can be avoided by preprocessing the data. In G-NAV this is done by a data precompiler. This program reads the user data and writes it again using a more efficient format that is perfectly compatible with the flight computer. Moreover, this tool can detect inconsistencies upfront, avoiding potential failures during operation.

TECHNICAL REMARKS

The G-NAV web application is a forked version of the native G-NAV and it is specifically developed and compiled for WebAssembly to run on web browsers. A minimal web service is required to load or install the app. Once installed or loaded, the application can run standalone without network connection. The web browser masters the application and it will keep it running for as long as it is being used. If the application goes to the background, there is no guarantee the browser will keep it in memory, as it might dump it in favor of other applications or efficiency.

Compatible web browsers

The application will run on web browsers where WebAssembly and WebGL is available, independently on the underlying operating system. Recent versions of mobile Chrome, Firefox and Safari have been tested, with all three of them being able to launch the application. The application has also been tested on desktop browsers Chrome, Firefox and Edge.

Each web browser comes with its own implementation of WebAssembly, so the perceived performance might vary, even when they run under the same operating system. It might therefore be useful to try different browsers when possible. The best performance has been observed in Android Chrome and iOS Safari.

It is highly recommended to update the browser before installing the app, as the support for progressive web applications is under continuous development.

Energy performance

Tests have demonstrated that the energy performance of the application depends greatly on four factors (some of which can be controlled):

- **Web browser**
  Each web browser comes with its own implementation of WebAssembly, and studies have demonstrated that these can result in considerably different energy performance.
- **Computational effort**
  G-NAV is designed to keep the amount of computations as low as possible by running simple algorithms and by reducing the computational frequency to 1 second. Still, some working modes are considerably more computational intensive than others. The S mode is the least intensive one, and R mode is the most intensive one. T is in-between, partially depending on the zoom level.

- **Size of bright areas**
  The energy performance is greatly affected by the size of bright areas. When displaying the terrain (either on the navigation screen or the route planner), life battery is expected to be reduced.
- **Screen brightness**
  The battery life is greatly dependent on the intensity of the screen brightness. In OLED screens, if a high screen brightness is set when displaying large bright areas (like the terrain), the battery life will be shortened considerably. Therefore, it is recommended to reduce the screen brightness when the application is on R and T modes.

Tips to save energy

If battery life becomes an issue, it is recommended to use modes R and T only during brief periods of time, as well as to decrease the brightness of the screen if possible.

If the screen is turned off, most browsers will pause the application timer and the geolocation dataflow, but they will resume everything when turning the screen on again. Experiment with this, and if it works well, it can be an effective method of reducing energy consumption when the application is not really needed. Take into account that some browsers might stop the application after some time of inactivity.

If there is no connection to the server while operating, stopping the application and restarting it later will only work if a proxy has been installed.

Energy performance example

This case serves only as a comparative example.
On a Motorola e6 mobile phone (3000mAh battery), running Android 9 and Chrome, the application will consume about 10% of battery per hour at 100% brightness in S mode (black background on navigation page). This means the total current driven by the application is about 300mA and the maximum expected autonomy is at most 10 hours. If the T mode is turned on, the consumption doubles even when the screen brightness is lowered to 50%, and the maximum expected autonomy is at most 5 hours.

Known problems

The next issues have been observed. If you notice issues that are not listed here, you can start an issue in GitHub.

Issue	Reason / Mitigation
After leaving the application in the background for several hours, it might fail to resume. This can be due to the loss of data (including the WebGL context).	Restart the application. Your current configuration is saved in the local storage and will be restored after restart. In Android Chrome, you can try to deactivate the default option of dumping

	the application memory when detected as unused. To do this, press the application icon, select ***App info*** and then disable the option ***Pause app activity if unused***.
No sounds are played (while the device is not muted).	First of all, the application must be installed in order to play the sound alerts. If still no sounds are played, check if the sound permission is granted. In Android Chrome, press the app icon and select ***Site settings***. Then enable the ***Sound*** option.
The screen dims off in Safari.	Safari does not allow wake locks, so you will have to configure the OS to keep the screen on while using the app.

GETTING STARTED

Providing a G-NAV navigation service

You can skip this full section and move to the section *Application life cycle* if you are not planning to provide a G-NAV web service, or if you are not curious about how G-NAV actually works in your phone.

Hosting the application

The application requires at least a static HTTPS web service to be loaded or installed. Note that browsers will refuse to start geolocation if the connection is not secure, so HTTP (without a security layer) is not sufficient. In the future we envisage having Ada-based web servers with SSL using the AWS library, but for the moment you will have to find your own alternative.

The application will request to the server the following files:

index.html	Contains the application HTML layout and the basic scripts that handle the timer, rendering, geolocation and user events and forwards the data to the WASM module.
main.wasm	This is the actual G-NAV WebAssembly (WASM) computational core module that manages all of the data and renders the screen using WebGL.
adawebpack.mjs	This file contains WASM binding to JavaScrip API's.
manifest.json	Contains information about the application, necessary for browsers to recognize G-NAV as a standalone PWA.
gnav-sw.js	This is the service worker, a standalone background process that is registered on the browser and interacts between the application, the browser and the server to provide all the necessary resources (i.e. all files in this table). This process is responsible for getting G-NAV to work when offline.

message.bin (N.A.)	This is a small file containing the last known message in human readable UTF-8 format. This is used to transmit brief, useful but not critical information from the server to all connected clients.
metar.bin	This is a small file containing the last known meteorological data from all the active METAR stations provided by the system in a precompiled binary format. Updates are requested to the server when necessary.
traffic.bin	This file contains densely packed air traffic information. Updates are requested to the server at regular intervals when the APRS function is enabled.
aircraft.bin	This file contains a list of aircraft in a precompiled binary format.
layers.bin	This file contains a list of map features (rivers, railroads, etc.) in a precompiled binary format.
airspaces.bin	This file contains a list of airspace sectors in a precompiled binary format.
reference.bin	This file contains a list of reference points and radio stations in a precompiled binary format.
terrain_1.bin terrain_2.bin terrain_3.bin terrain_4.bin terrain_5.bin	These files contain the topographic information in a precompiled binary format.

A service is only intended to cover a limited extension of the planet (due to limited data volume). If your area exceeds the limits, consider splitting your area in different regions and to assign a service for each region.

Service information data

Some basic service information needs to be provided on the header of the index.html file. The reference position is here mandatory, as it is required to determine how geographic locations will be internally stored in the GPU buffers. This is essential to avoid excessive

distortion from the non-conformal mapping and to increase performance by reducing the amount of operations necessary to load the data.

Elaboration of data files

Data files need to be precompiled in a binary format using *gnav_crunch*, the G-NAV data pre-preocessing program (which can be created for Windows, Mac and Linux using a standard Ada compiler). The data compiler is a command line tool that takes input files in a standard and/or human readable format and translates them into a more G-NAV friendly format. This not only ensures compact files, but it also simplifies the data structure, which makes the transferring and loading much more efficient than using the original files.

Elaboration of terrain data files

The original terrain file needs to be acquired in ESRI grid format. This can be extracted from Open Topography or similar services and processed directly by the data compiler.

ESRI grid files come with a constant cell size in both latitudinal and longitudinal directions, which means that closer to the hemispheres cells will have a higher aspect ratio. This is undesired for G-NAV, since there is no use in having different resolution on each screen direction. Therefore, the number of cells in each direction should be controlled to obtain a cell aspect ratio as close as 1 as possible. For example: at 60° in latitude, the aspect ratio of a constant cell will be 0.5, which means that the longitudinal cell size can be doubled. This will produce square cells in a local isometric plane, which is better adapted to a screen.

Controlling the cell size will also help decrease the file size (and resolution) when necessary. The data is internally limited to a rectangular grid of 10 million elevation points. Each data point is encoded in 2 bytes, resulting in a maximum of 20 MB of data. This data bound means that the maximum horizontal resolution of the terrain is inversely proportional to the area of the target region.

The ESRI terrain file must be compiled using the G-NAV data compiler by passing the TERRAIN argument. The compiler will search for the files terrain.bin or terrain.dat (in that order) inside the files folder, process it and then distribute the result evenly in five different files named terrain_N.bin where N goes from 1 to 5.

Elaboration of the layers data file

The map layers can be built using ESRI shape files and the G-NAV data compiler. The name of the files is rigorous:

- CC_rivers.shp
- CC_lakes.shp
- CC_rails.shp
- CC_roads.shp
- CC-CC_border.shp

Note that all names are prefixed by CC or CC-CC, where these two characters denote a country code (for example: DE, BE, AR, AU, etc). This code can be anything, but it must be consistent.
NOTE: there is a limit to the number of entities that can be loaded. The data compiler will inform you about this.

Elaboration of the airspace data file

The airspace sectors must be listed in a standard *open air* file named airspaces.air. This format is ASCII based. The compiler will process the file and generate a binary version when called with the AIRSPACE argument. Airspace sectors will be constructed and broken in smaller parts where necessary to increase the performance at the client side. Note that portions of the airspace sectors that fall outside the coverage area are trimmed.

NOTE: there is a limit to the number of entities that can be loaded. The data compiler will inform you about this.

Elaboration of the reference data file

Part of the reference points are listed in an ASCII UTF-8 text file named reference.dat. Each line contains a single reference point and must be formatted as follows:

```
#<KEYWORD>
<4 letter Id> @ <14 letter name> @ <N/S>DD*MM'SS.S" <E/W>DDD*MM'SS.S"
```

All letters must be upper case. The allowed keywords are:

```
#WOODS, #LANDMARKS, #WATER, #TURBINES, #AIRFIELDS, #VILLAGES, #CITIES,
#LANDOUTS
```

For example:

```
#WOODS
BRKB @ BRAKELBOS   @ N50*46'16.7" E3*43'24.3"
```

Airfields and cities can also be provided by CSV files (see the source code to understand how this is done).
Radio frequencies can be loaded using a custom radio.dat file containing the station names and associated airfield ICAO codes when applicable.
The data compiler processes the data with the REFERENCE command and returns references.bin when started with the REFERENCES argument.
NOTE: there is a limit to the number of entities that can be loaded. The data compiler will inform you about this.

Providing real time METAR

NOTE: this feature is under development in the current version.

When METAR=TRUE is set on the configuration part of the HTML file, the application will attempt to fetch the file metar.bin from the server when necessary. The content of the file is packed in a binary structure that the G-NAV server creates using information from the connected to METAR message providers.

- 8 bytes: message timestamp in seconds since the epoch (double)
- 4 bytes: lapse from the timestamp to the next message in seconds (float)
- 1 byte: number of meteo stations in this block (unsigned)
- Per meteo station, a block of 18 bytes:
    - 4 bytes: unique identification string (ICAO name)
    - 4 bytes: age relative to timestamp in seconds (float)
    - 8 bytes: latitude of the station in degrees (double)
    - 8 bytes: longitude of the station in degrees (double)
    - 1 byte: QNH offset to 1013hPa (signed)
    - 1 byte: wind speed in km/h (unsigned)
    - 2 bytes: wind direction in degrees (signed)
    - 1 byte: temperature in °C (signed)
    - 1 byte: dew-point °C (signed)
    - 2 bytes: cloud based in m (unsigned)
    - 2 bytes: visibility in m (unsigned)

Providing real time air traffic

NOTE: this feature is under development in the current version.

When TRAFFIC=TRUE is set on the configuration part of the index.html file and the APRS function is enabled, the application will attempt to fetch updates of the file traffic.bin from the server at regular intervals (normally 6 seconds). The file contains a reference location and a timestamp followed by a list of recent tracks in the vicinity of the reference.

- 4 bytes: timestamp in seconds since 00:00 Zulu
- 8 bytes: reference latitude
- 8 bytes: reference longitude
- 1 byte: number of tracks in this block (unsigned)
- Per track, a block of 18 bytes:
    - 4 bytes: unique identification (OGN hexadecimal identification code)
    - 1 byte: age relative to timestamp in seconds (float)
    - 4 bytes: latitudinal offset in degrees (float)
    - 4 bytes: longitudinal offset in degrees (float)
    - 2 bytes: altitude AMSL in m (unsigned)
    - 1 byte: vario in multiples of 0.1m/s (signed)
    - 1 byte: speed in multiples of 2 m/s (unsigned)
    - 1 byte: course in multiples of 1.5 degrees (unsigned)

The G-NAV project provides an OGN client that is able to connect to the APRS servers and generate this file automatically every second. The program checks for the consistency of the data and is able to filter tracks per target kind and location.

Application lifecycle

Installing

In principle, no installation is required to run G-NAV. When loading the application for the first time, a *service worker* will be registered in the web browser and the static application data will be locally cached on the browser. The application will be able to run offline in that browser as long as the service worker and the cached data are available. Most mobile browsers allow installing PWA's to improve the user experience by providing a direct access icon on the main screen. In Chrome, for example, you can do this by selecting the three dots, and then choosing *add to main screen* and then *install application.*

Updating

To update G-NAV in Android Chrome, close the app, maintain the icon pressed, select ***site settings*** and then select ***delete data and reset permissions***. Then restart the application.

Uninstalling

Uninstalling the application can best be done by removing the cached data from the browser. This can usually be done from the *privacy and security* configuration settings in the web browser. Another way to uninstall the application is by following the standard procedures provided by the operating system (iPhone or Android).

Alternatively, the app can be uninstalled by dragging the app icon to the remove icon.

Storage requirements

The application only requires a minimum amount of storage space:

- Approximately 1MB for the WebAssembly module (execution module).
- Not more than 30MB of data.

Local storage

The application will automatically register data in the local browser storage only to keep the user configuration active (flight plans, selected aircraft and waypoint, wind and other user data). By doing this, the application can reestablish its last state and be resumed after a shutdown.

The next data is being saved on the local storage. You can locate the keys and values using your browser development tools.

- AIRCRAFT: the current aircraft registration.
- PLAN: the current flight plan index.
- WAYPOINT: the current waypoint index.
- PLAN1 to PLAN10: name and the waypoints of each loaded flight plan.

Debugging

While developing the service, it can be useful to check the log messages thrown by G-NAV in the browser console. Additionally, most browsers contain sophisticated development tools that will let you see what the application is doing.

Privacy and security

The G-NAV client web application requires secure HTTP to work, because it needs access to your device's position. You will be asked for permission to switch on the geolocation module, which is a standard function of the browser.

In the current version, G-NAV downlinks information back to the web-server when the APRS mode is enabled. This mode is disabled by default, and the information that is transferred consists in the position, altitude, speed and course. This is necessary to provide customized traffic information and logging service.

Only connect to G-NAV services that you can trust. The risk of launching web services of unknown precedence is that these might include extra code and mask a different purpose (like collecting or stealing data).

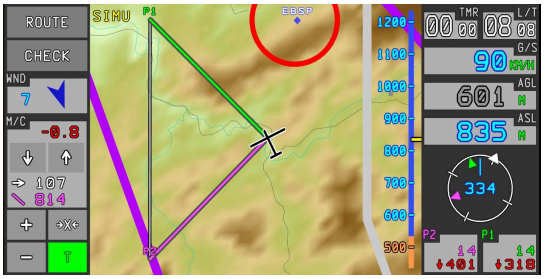
FEATURES AND FUNCTIONS

General organization of the program

G-NAV strongly prioritizes safety and efficiency over flexibility, and for this reason, you cannot modify the color or position of the components. The information is always at the same place and it means the same for everyone. So if you take the device of your friend, the application will look exactly the same and you will not have to readjust your thoughts.

Screens

G-NAV is based on a main navigation screen containing a functions column at the right, the map viewport at the center and the flight data at the right. The functions panel is at the left side so that it can be reached without covering the screen when the device is at the right side of the user. Some buttons on the functions panel provide access to secondary screens.



- ROUTE > ROUTE_EDITOR
- CHECK > INFO, GLIDER, WIND, SYSTEM

The navigation between the screens has been designed to be as fluent as possible: there is in general only one level of sub-pages (only the route edition page is a level two page) and all screens other than the navigation screen provide a red NAV button on the top left corner to go straight back to the main navigation screen.

Aircraft data

General

G-NAV can hold up to 10 different aircraft. The selection is done in the GLIDER panel (on the CHECK page, the name of the selected glider model is visible as a tab button).

The selected glider is the one used for all performance computations (range, optimal speed and required altitude), so it must be chosen immediately when starting the application. Using the application with a wrong model will result in incorrect predictions.

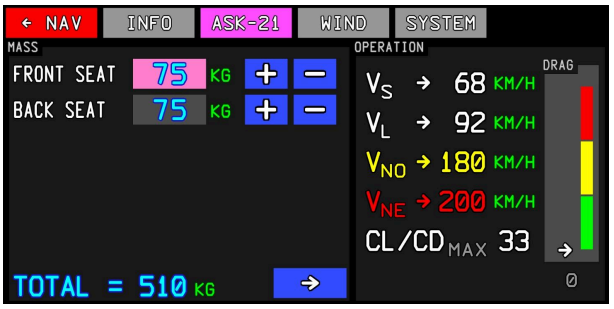
The selected aircraft is saved in the local storage and restored the next time the application is launched. The general configuration of the aircraft is published and maintained by the service provider and cannot be adapted from the application.

Mass

The total mass of the aircraft is used for limit checks and performance computations, so make sure it is correctly entered. If the mass of an occupant or ballast exceeds the maximum allowed mass, the entry will blink in red. If the total mass exceeds the maximum take-off weight, the total mass will blink in red.

Performance

The performance of the aircraft for the given mass is measured by splitting the drag in an induced component and a parasitic component. The parasitic drag component is obtained as a combination of a clean state drag and a rough state drag using the drag slider. The green area of the slider goes up to about halfway between both drag states. On the red area the drag is fully rough.



The clean state drag is the drag obtained from the original polar curve after subtracting the induced drag component, which is approximated using the Prantl formula (this is why the effective aspect ratio is necessary as data parameter). The rough drag is not strictly defined, but it is recommended to use experimental data for a similar airfoil (or glider) under standard roughness.
The clean state drag is evaluated using a fourth degree polynomial on the lift coefficient (CL). The coefficients of the polynomial can be obtained from the original polar using the provided Scilab scripts that are part of this project. The rough state drag is given by a second degree polynomial on the CL.
The CL/CD MAX value displayed on the GLIDER panel is the maximum gliding ratio for the given drag combination without sink nor wind, and it is very useful to have an idea of the resulting level of degradation when moving the drag slider.
Some performance curves used by G-NAV can be found in the repository. These images have been generated using Scilab.

Home and route planning

Home

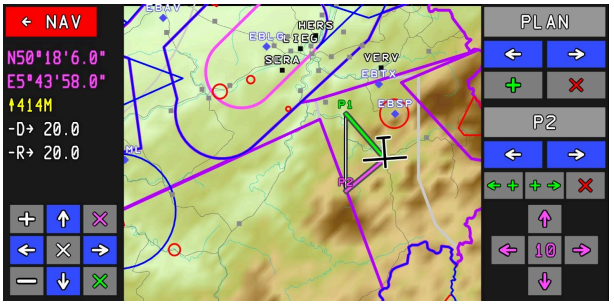
The application always keeps a visible vector to the home location, which is assumed to be the first waypoint in the route. The home waypoint navigation data is displayed in green at the lower right corner of the navigation panel. This info panel is split in two by default, showing the home and the active waypoints. Touching the panel will remove the home waypoint and display extra information about the active waypoint. This is very useful for when there is only one navigation point (the home waypoint).

Waypoints

You can build a route by specifying different waypoints. G-NAV can hold up to 10 different routes and each route can hold up to 20 waypoints. The selection and edition of the active route is done from the ROUTE page using the arrow buttons.

To change the name of a route or a waypoint, click their corresponding names and a keyboard will become visible. Click elsewhere on the screen to finish the edition.

To erase a route or a waypoint, click the red cross buttons. Note that this operation always requires confirmation to minimize the chance of removing data accidentally.



To add a route, click on the green plus button. There is no confirmation required and the route will always be added at the end of the stack.

To add a waypoint, click either on the left or right green arrow buttons. The waypoint will be inserted before or after the current waypoint.

The active waypoint can be moved by steps of 10km, 1km or 100m. To change the step size, click on the button at the center of the moving arrows. These steps are sufficient to move the nodes very fast at any point with a very decent accuracy.

In the ROUTE page you can move the view using the arrows or by pressing the different X buttons, which will bring you to the home or active waypoints, or to the current position as indicated by their colors.

Routes are saved in the local storage and will be restored after restarting the application.

Active waypoint

There is always one active waypoint and the associated data is always displayed in magenta. The active waypoint will automatically change when the position of the aircraft is within 1km from the current waypoint. The next waypoint will either be the next one or the previous one, depending on the status of the backtrack function.

The selected waypoint is restored after restarting the application.

Backtrack function

The BACK button in the WAYPOINT page is used to toggle the backtrack function. When this button is highlighted, the route is followed in the reverse order. This means that when you approach a waypoint, the previous one on the table is activated.

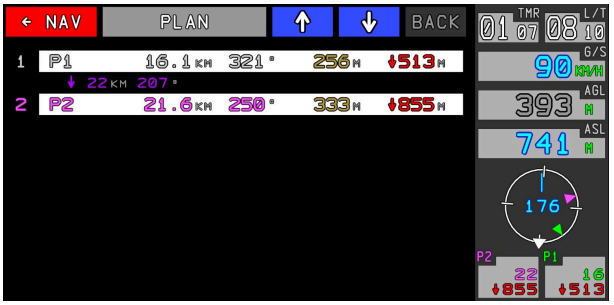
Manual selection of a waypoint

The active waypoint can be changed manually on the WAYPOINT page or on the ROUTE page using the blue arrows.

It is not possible to select a waypoint that is within 1km range from the current position, as the automatic proximity detection will override the selection.

Route follow-up

The route can be followed up in details on the WAYPOINT page. The strips on this page provide information about each waypoint (distance, course, local elevation, arrival altitude). In between the strips information is given about the leg between two waypoints (distance and course).



Flight data

Generalities

G-NAV will build-up functionalities based on the available data. If some data is missing, becomes too old or is invalid, functions requiring that data will be disabled and dashes will be displayed instead of the last known value. For the ground speed, altitude, elevation and course, the lifespan of an update is no more than 2 seconds.

Clock

A clock can be found at the top of the navigation panel (in the NAV and WAYPOINT pages). The clock gives the hour in either local time (L/T) or in UTC, as indicated by the frame label. These two can be switched by clicking on the frame.

Timer

A timer can be found at the top of the navigation panel, left of the clock.

The timer can be reset by pressing two times on its frame. The first click turns the TMR label in red, indicating that the second time the frame is clicked the timer will be reset.

The timer starts counting in minutes and seconds for the first hour, and then it switches to hours and minutes. For energy-saving reasons, the nominal screen refresh rate is set to 1Hz, so when the timer counts in seconds, it might happen that some seconds appear to last longer.

The timer is stored locally, so if you restart the app, it will keep counting from the moment it was reset.

Position

The current position is indicated in the moving maps (visible in the NAV and ROUTE pages) with a simplified glider-shaped icon, and it is also printed in geographical coordinates using sexagesimal notation at the top of the CHECK page. The position is normally updated every second when there is good reception. After two seconds of not receiving positional data, a red blinking GNSS label will be displayed at the top-right corner of the moving map and the coordinates in the CHECK > SYSTEM page will turn gray.

Ground speed

The GNSS derived ground speed is indicated in the navigation panel inside the G/S frame. The units can be toggled between km/h (default) and kts by pressing the frame.

Always remember that this is not equal to the airspeed.

Altitude

The GNSS derived altitude (above sea level) is indicated in the navigation panel inside the ASL frame. The units can be toggled between meters (default) and feet by pressing the frame.

The altitude units are coupled to the elevation, the altitude look-up table in the CHECK > INFO page and the sector vertical limits displayed in the moving map.

The provision of altitude from the mobile system requires the view on more satellites than those required for the position and speed. Therefore, it is possible that the altitude field

fluctuates or becomes unavailable during some intervals. In the latter case, some functions will be degraded.

When the altitude is available, the altimeter bar is displayed. The graduation of the bar is in hundredths of meters, starting from the MSL. Below mean sea level, the bar becomes gray.

Elevation

The calculated elevation (above ground level) is indicated in the navigation panel inside the AGL frame. The units can be toggled between m (default) and feet by pressing the frame.

This field is internally calculated using the altitude and the underlying ground elevation, so it requires the presence of a terrain grid.

When the elevation is available, the altimeter bar will display an orange strip for the first 300m from ground level. Below the ground level the bar becomes gray.

Trajectory

The trajectory of the flight is always represented on the moving map. The polyline is extended when the position is updated for more than 100m.

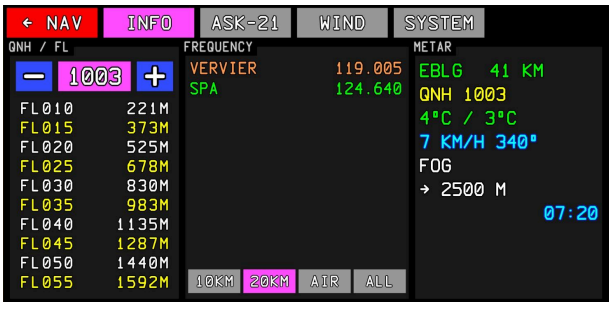
After a jump in the position of more than 80% and over 100m, the polyline is broken and the variation of the position is monitored by measuring the difference in distance covered by the last three updates. When the difference is under 50%, a new polyline is started (disconnected from the previous one).

The trajectory has a limited size.

METAR

If the web service provides METAR information, these messages can be found at the bottom-right of the CHECK page (within the METAR frame). G-NAV checks for METAR updates every 15 minutes and notifies the time when the last fetch was last done.

When a recent METAR is provided, the QNH and wind data can be manually extracted from it. G-NAV does not automatically extract data from METAR messages.



QNH

The QHN can be entered in the CHECK page.

The QNH can be active or inactive. The default state is inactive. When it is activated, the vertical limits of the sectors on the moving map are converted from FL to altitude where applicable.

Radio frequencies

In the CHECK page a list of radio frequencies is provided. The buttons 10 and 20 can be used to filter the stations that are within 10Km and 20 Km respectively. The AIR button can be used to filter the stations that do not have a declared airfield.

Real time traffic - APRS

When real time traffic is provided by the server and the APRS function is enabled, G-NAV will attempt to fetch surrounding tracks at regular intervals of 6 seconds and represent them on the map.

The APRS function can be controlled via the APRS panel located at the CHECK > SYSTEM page. The TRAFFIC toggle button serves as a master switch. The function is active when the button is drawn in magenta.

By default, when the APRS function is active, only the position of the selected home waypoint will be included in the header of the XHR request that is sent to the server. The server can then use this position to send only the traffic around this location (the radius is configurable at server side).

To receive the traffic surrounding the actual position, the FOLLOW mode must be activated. This will also include the altitude, speed and course in the request as an IGC message. The server can record this data and use it to enrich the traffic data.

If you notice the APRS system is not providing much useful information or it is distracting you, turn it off.

NOT ALL TRAFFIC IS DISPLAYED ON THE SCREEN. The idea of the function is to make you aware of traffic that you probably did not see yet. Do not spend too much time looking for the traffic that is on the screen unless you see there is a real risk.

The symbol used to represent traffic is the same as for the own glider, but smaller. Two colors are used:

- Yellow: tracks not older than 10 seconds since their emission.
- Pink: tracks older than 10 seconds since their emission.

The emission is the moment at which the data is sent from the reporting unit (i.e. aircraft), not the moment at which the G-NAV server received it.

Yellow tracks are extrapolated using their last known speed, course and rotation rate, with this last variable being faded over time.

Tracks older than 10 seconds are frozen at the last extrapolated position. Tracks older than 30 seconds are not displayed, and normally also not provided by the server.

When APRS is enabled, the status is displayed on the right top corner of the moving map in the navigation page:

- APRS in green: the system is receiving fresh traffic data from the server.
- APRS in yellow: the system is receiving outdated data. This means that either there are no tracks or the server is having issues. These could be local processing issues or disconnection from OGN.
- APRS in red: no traffic data is being received. This means that either the server is down or a connection could not be established.

Geographic data

Air traffic sectors

The moving map will display the loaded sectors in a fixed order per sector category. Each category has a different color and line thickness assigned:

- CTR: Magenta
- TMA: Blue
- CTA: Blue
- RSA: Red
- ? Orange
- G-airspace: Purple

There are configurable sector incursion and or excursion warnings in G-NAV. Still, we encourage you not to depend on them and to carefully study the flying area before takeoff and to follow the situation visually on the moving map.

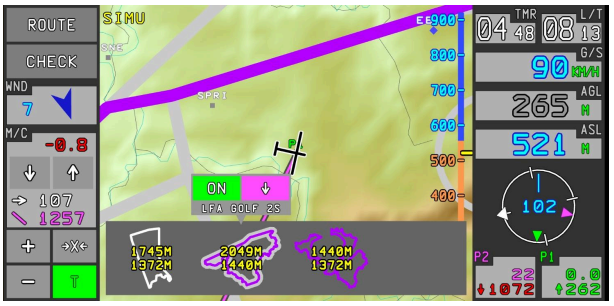
Selection

To study and configure the map sectors on G-NAV select a location in the map by briefly pressing the screen. If there are sectors containing the selected location, a horizontal list will appear. Select the desired sector by pressing on it. If the sector is not visible on the first page, swipe to the right and to the left to change the pages.

When you select a sector, the boundary of all other sectors is temporarily displayed in gray and the sector name and status is displayed on a vignette.

Configuring the status

To change the status, press on the sector info vignette. When alerts are turned off, you can only change the status between active or inactive. When alerts are turned on, you can configure the kind of notification you want to receive for each sector: when entering (arrow down), when leaving (arrow up) or both cases. Take into account that the status of the sectors is updated when the location changes more than 250m.



After closing the sectors list, the status of the sectors is stored locally so that it can be restored the next time the application is started.

If the sound is allowed, the application will emit a single high pitch beep sound every time a configured sector incursion or excursion is detected. To check if the application sound is working, G-NAV emits a single beep at startup. If you don’t hear that beep, it is because the host browser is not granting the sound permission.

Inactive sectors will still be displayed everywhere, but in light gray. It is best practice to configure the status of each sector where you are planning to fly prior takeoff using the most recent NOTAMS, as doing this during flight will become a source of distraction.

Airfield frequencies

If an airfield is linked to a radio station, touching the airfield will display the associated radio frequencies.

References

G-NAV can display the following references in the moving maps:

- Airfields

- Wind turbines
- Cities
- Woods
- Roads
- Railroads
- Rivers

Wind and estimated range

Wind entry modes

The wind provided to the system must be the *expected average wind*, which is not necessarily equal to the current local wind. When gliding down over a long extension of land the wind might change in time, location and altitude, so in absence of accurate information, the most suitable wind value for a range estimation is an average. This is why G-NAV does not force the usage of the calculated wind, but rather provides it as a suggestion.

Manual wind mode

When the MANUAL function is on (the button is highlighted in magenta), the system expects the pilot to enter a suitable wind. The wind can be set by clicking on the WIND panel in the CHECK window. Simply press on the screen to adapt the direction.

Metar wind mode

When the wind is in AUTOMATIC-METAR mode, the wind provided by the metar is loaded as soon as the metar is updated.

Calculated wind mode

NOTE: this function is under development and it has only been tested using a flight simulator. The current version does not provide this option yet.

When the flight enters in circling mode, G-NAV estimates the wind based on the drift. This value can be adopted automatically when the AUTOMATIC-COMPUTATION function is on.

Range estimation functions

G-NAV makes several important computations based on the provided wind, expected sink-rate and the aircraft performance. One would tend to oversimplify the problem by only looking at the minimum gliding slope, but this performance is only achievable in a perfectly stagnated atmosphere. The estimation of the range is more complex than that, as it involves many variables, some of which cannot be calculated and must be provided based on experience and the level of willingness to assume risks.

The air in the atmosphere is seldom stationary, and while flying, the aircraft will encounter areas of upwards and downwards wind. Since the strength and extension of these areas is

impossible to predict accurately, there is no way to accurately predict the range either. If you are good at studying the clouds, you'll be able to avoid areas of downwind and increase your range considerably, even when you don't take the shortest path between points along the route.

You are the only sensor able to assess the atmospheric conditions and there is no way to transmit all these experiences and thoughts to the computer. So in G-NAV the problem is reduced to something simple and practical. Nonetheless, you will still need some training and practice if you intend to use these functions. Basically, you will have to learn to adjust the open variables to study the best and worst outcomes, and to assume a level of risk. You will have heard from experienced pilots that gliding is all about confidence. Then, to minimize risk, make sure your confidence is based on a good perception of reality and that there is always a good outland alternative for the worst outcome.

Expected vertical wind component

The expected vertical wind component (MC value) is the average downward stream velocity of the air relative to the ground that one expects to find along the way. The value is always indicated in m/s. This is a critical variable because it will strongly affect the result of all range computations, as much as the horizontal wind or even more. The value must be entered using the arrow buttons that are always visible on the M/C panel in the NAV page. These buttons have been placed there in order to have direct control on the variable and being able to try different scenarios during the flight.

The evaluation of an appropriate MC value starts with a pre-assessment of the weather conditions: strong thermals are normally paired to strong sink. Here you need to evaluate how strong the sink can be along the intended way, and how certainly these conditions will be met along the intended way.

Starting from there, a level of optimist must be assumed based on:

- **Intended gliding distance**: the shorter the glide, the more the outcome will be affected by the vertical motion of the air, and it is very unsafe to assume an optimistic value without being certain about the coming conditions. In a long enough glide (several times longer than the space between thermals), the upwards and downwards motion of the air will tend to neutralize, and it is safer to assume an optimistic value.
- **Current altitude above the ground**: due to the same reasons, the optimism should be reduced at lower altitude, since encountering a strong sink at low altitude is much more critical than at high altitude.

The next conditions give an indication of how to select a suitable MC based on the assumed level of optimism:

- **Between 0.0 and -0.5: optimistic**
  You are gliding down a long leg at a safe altitude, or you know exactly that this value is representative for the real conditions. Note that as a safety feature, if you

encounter or expect upwards currents along a large part of the way, G-NAV will not let you improve the range more than that given for 0 MC.

- **Between -0.5 and -2.5: neutral**
  You are still hesitating about the conditions along the way, or you intend to glide a short leg (for example, a distance less than the space between three thermals) but you have a safe altitude above the ground.
- **Between -2.5 and -5.0: pessimistic**
  You can't predict the conditions, or you are certain that there are high chances they might turn this bad and/or you intend to fly at not too high altitude.

Tactical tip: an important recommendation is not to increase the optimism at the cost of accepting a high risk. Be conscious about your level of optimism and always fly with a second plan in mind. For this, toggling different MC values can be useful: you fly optimistically towards your goal, but knowing that in the worst case a suitable out landing field is within reach.

Sink warning

The system will continuously monitor the state of the flight. When a long enough block of relatively straight trajectory is detected (see data performance register), the average gliding slope is compared against the best outcome for the provided MC. When the measured slope is above the computed one, the arrow button pointing downwards on the M/C panel will shine in magenta, indicating that the given value is too optimistic at current conditions. The user should assess the situation and eventually press the button until the alert is gone or at least inactive during most of the flight. This will reduce the expected range to a more realistic value.

Note that the alert is susceptible to changes in the flight direction if the provided wind is too inaccurate. Also, the system assumes that the airspeed is being kept at the given optimal value (resulting in the smallest possible glide slope).

If the sound is allowed, the application will emit a single low pitch beep sound every 15 seconds while the warning is active. To check if the application sound is working, G-NAV emits a single beep at startup. If you don't hear that beep, it is because the host browser is not granting the sound permission.

Estimated descent and optimal speed

G-NAV computes the estimated descent for the next waypoint in straight flight based on the aircraft performance, wind direction and MC. The value is an estimate because it is affected by all inaccuracies inherent to the provided data (polar curve mismatching, wind and MC uncertainties). And on top of this, the estimation is also only valid if the flying airspeed is kept at the provided optimal value.

In G-NAV, the provided mass truly affects the performance of the glider, so it is important to introduce accurate mass values. Therefore, remove the weight of the passenger when flying solo in a double-seat aircraft.

The estimated descent and optimal airspeed are indicated on the lower-right corner of the moving map, left of the ASR function.

Estimated altitude at arrival

On the WAYPOINT page, G-NAV provides the estimated altitude at arrival for every waypoint at the current conditions in straight flight.

Estimated geographic range cones

This function is explained in the next chapter.

Display modes

S-mode

In S-mode, only geographic references are displayed. The topographic chart is turned off. This is the least-power consuming state of the application.

T-mode

In T-mode, an adapted topographic representation of the area is displayed on the moving map. In this mode, this layer only serves as a visual reference.
Take into account that activating this mode will increase the power consumption.

R-mode

G-NAV uses the topographic data and the range estimation algorithms to represent the areas that are within maximum range in a straight line at optimal airspeed.

NOTE: if an obstacle is between two unshaded areas in the form of an island or peninsula, the area behind is not within straight-line range, and probably also not within range at all. If the area is in blue, take into account that the margin will be below the safety buffer.

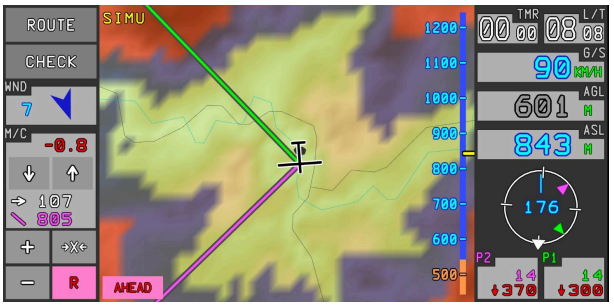
Take into account this mode is the most power consumption state of the application, so you might want to limit its use in some circumstances.

The R-mode can work in three different ways:

- AHEAD
  The range is the maximum straight ahead, based on the provided configuration and flying at optimal speed. The areas where the altitude reaches the ground at arrival are shaded in red, and the areas where the altitude is under 200m above the ground at arrival are shaded in blue.
- LOCAL OPTIMAL
  The interface between the red and the blue areas indicates the flying boundaries where the active waypoint is reached at ground level, and the interface between the

blue and the unshaded areas indicates the flying boundaries where the active waypoint is reached at 200m above the ground.

- LOCAL 10:1
  The interface between the red and the blue areas indicates the flying boundaries where the aircraft will intersect a 10:1 cone that starts at ground level from the active waypoint, and the interface between the blue the unshaded areas indicates the flying boundaries where the aircraft will intersect a 10:1 cone that starts from 200m above the ground from the active waypoint.

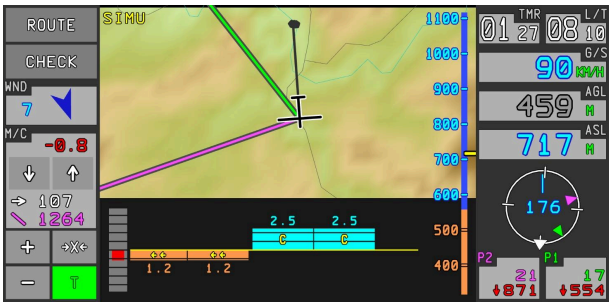


Flight performance registers

G-NAV will summarize the most recent flight data in no more than 10 blocks of time that can be between 10” and 1’30” long. The system will only register blocks that can be classified under one of the following maneuvers:

- **Straight path**: the path was kept more or less straight during the registered period. This means that a minimum turn rate of 2°/s was established at the beginning and the turn rate did not exceed 6°/s during the measured time span. The average sink rate (in m/s) will be displayed.
- **Circling path**: the trajectory does not classify as straight, a turn of at least 360° was established and a given turn direction (clockwise or anticlockwise) was kept during the measured time span. The average lift rate (in m/s) will be displayed.

The summary of the registers can be toggled by touching the ASL data panel.



In both cases, the average vario will also be displayed next to the sink or lift. Transitions (maneuvers) are not registered because they do not provide very useful information.

Annex

Theoretical frame behind the range estimation

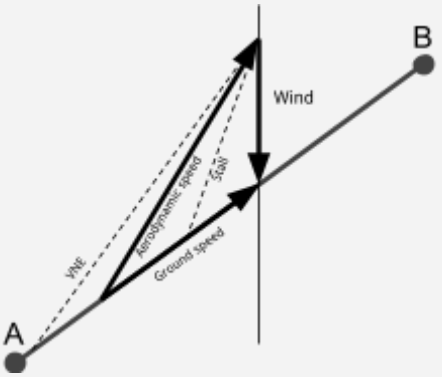
The range estimation is done by analyzing the aircraft performance in a given direction using the provided wind and sink rate, with the motion of the atmosphere assumed to be perfectly uniform along the path. This means that the air can be thought of as a reference frame (refer to textbooks in kinematics -as for instance R. A. Tenenbaum- for more information about what a reference frame means).
The aircraft performance is provided by a set of dimensionless lift and drag force coefficients (the usual CL and CD) at steady state conditions. It is assumed that this state is representative for any position the CG might adopt. In other words, the influence of the elevator deflection in the overall drag is neglected. Furthermore, in the current version only one polar set is allowed, so different flap configurations are not possible unless declaring them on a separate aircraft.
The program finds the minimum gliding slope by scanning the full polar range, that is to say, from the stall to the never exceed airspeeds.
The equilibrium of forces for the provided weight at current altitude resolves into the gliding angle and the aerodynamic speed for each polar state. The gliding angle is given by the next equation,

$$\gamma = atan\left(\frac{C_D}{C_L}\right)$$

and the aerodynamic speed is approximately given by this other equation:

$$V = \sqrt{\frac{2\,g\,M\,cos(\gamma)}{\rho\,S\,C_L}}$$

The exact solution for the speed is actually more complex and requires the usage of numerical methods. However, for practicity it makes sense to take the above solution. The aerodynamic velocity vector (the velocity of the glider relative to the air mass) can then be easily decomposed in vertical and horizontal components. Then a vector composition of the horizontal aerodynamic velocity and the provided wind gives the resulting ground speed and the trajectory correction angle.



Finally, the gliding slope is obtained by the ratio between the rate of descent and the ground speed, where the rate of descent is the composition of the vertical component of the aerodynamic speed and the vertical component of the air mass.
The gliding slope turns out to be a multivariable function dependent on the course, altitude (through the air density), weight, wind and sink rate.
If you are curious to know how this works in practice, take a look at the procedures ***Calculate_Gliding_States*** and ***Calculate_Gliding_Spectrum*** located in the ***Flight.Aircraft*** package of the source code. You will note that in the numerical implementation, the calculation is optimized by using the wind direction as reference, not the intended course, since this one is variable. By doing this, the program only needs to recompute everything when at least one of the dependent variables changes considerably, regardless of the trajectory.
The omnidirectional set of gliding slope lines form a cone when traced from the aircraft position. The intersection of this cone with the topography is an estimate of the maximum range straight ahead.
There is however another problem that is equally important: the local range estimate. Here the question is not how far away you can go ahead to reach the ground, but rather how far I can go and safely return to a given point, or to stay inside a given cone centered on it.

The mathematical problem is quite similar, but requires two steps.