G-NAV

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

March 2024
Release 1.0

Evaluation and demonstration stage

Preface

G-NAV is a progressive web application (PWA) for air navigation, developed under a non-profit organization of volunteers. 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.

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 an optional complement based on a different sensor, namely the GNSS.

A lot of features can be included in a 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 high, the product becomes too hard to

maintain and operate. A limited number of features, on the other hand, results in a more stable and resilient product.

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 so as to avoid source code that could lead to bugs.

These are some of the design rules and guidelines that are being consciously followed in the development of the application.

- 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 the same color.
  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 too much resources and complexity, while most of the time they provide little to none added value in a flight computer. 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 deliberately free of these features.
- Functions must be readable and to the point. If something isn't readable it will only consume more time for the pilot to interpret.
- 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.
- 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. This is called smooth degradation.
- 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.
- 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.

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 can also be launched on a desktop browser.

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.

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. 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 use case

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.

Knows problems

These issues have been observed and remain unsolved.

- After leaving the application in the background for a period of a few hours or more, it can fail to resume. Mitigation: restart the application. Your current configuration is saved in the local storage and will be restored after restart.
- After a long period of several hours, the wind page might fail to render correctly. This is likely due to an issue in the custom calendar that needs to be fixed.

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 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 the 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.dat (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.dat	This is a small file containing the last known METAR data in human readable UTF-8 format. Updates are requested to the server every minute when necessary.
traffic.bin	This file contains densely packed air traffic information. Updates are requested to the server every 8 seconds.
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.
reference.bin	This file contains a list of reference points 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 the G-NAV data compiler (which can be created for Windows, Mac and Linux using a standard Ada compiler). The data compiler 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.

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).

Elaboration of the airspace data file

The airspace sectors must be listed in an ASCII UTF-8 text file named airspace.dat. The compiler accepts point to point lines in WGS-84 coordinates, and also boundaries, arcs and circles, which is the way sectors are typically defined by ANSP’s.

The airspace data file are compiled along with the layers using the LAYERS argument, because the sectors are usually referenced to border lines. These border lines must be provided as ESRI shape files.

The format of the airspaces.dat file is as follows:

```
#<Comment>
<20 letter name>
<Kind>/<Class>/<Lower limit>/<Upper limit>/<Label location>
<DDMMSS><N/S> <DDMMSS><E/W>
CIRCLE <Radius> <NM/KM> <DDMMSS><N/S> <DDMMSS><E/W>
ARC <Radius> <NM/KM> <DDMMSS><N/S> <DDMMSS><E/W> > <DDMMSS><N/S> <DDMMSS>
BORDER BE-NL
```

For example:

```
######################
#FIR
######################
BRUSSELS LOWER CONTROL AREA
FIR/C/4500FT/FL195/505902N 0031110E
510521N 0023244E
510700N 0020000E
513000N 0020000E
512223N 0032147E
BORDER BE-NL
...
510521N 0023244E
```

Elaboration of the reference data file

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"
```

The data compiler processes the data and returns references.bin when started with the REFERENCES argument.
A maximum of 500 reference points can be loaded.

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 index.html file, the application will attempt to fetch the file metar.dat from the server every minute. The content of the file will be displayed on the METAR panel located in the CHECK window. The number of characters here is restricted to two lines of 35. Words are not cut in the middle, they are carried to the next line if they do not fit.
METARS are normally produced every 30 minutes. After the reception of the first METAR, G-NAV will extract the timestamp, pause the request and resume it when the next metar is expected.

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, the application will attempt to fetch updates of the file traffic.bin from the server every 8 seconds. The file contains a timestamp followed by recent tracks.

- 8 bytes: timestamp since the epoch (1/1/1970 00:00 UTC) (float)
- 2 bytes: number of tracks in this block (unsigned)
- Per track, a block of 18 bytes:
    - 4 bytes: unique identification (OGN hexadecimal identification code)
    - 2 bytes: age relative to timestamp as (float)
    - 4 bytes: latitudinal offset in degrees as (float)
    - 4 bytes: longitudinal offset in degrees (float)
    - 2 bytes: altitude AMSL in m (unsigned)
    - 1 byte: speed in m/s (unsigned)
    - 1 byte: course in degrees (unsigned)

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*.

Local storage

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

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 between shutdowns.

Updating

In the current version, updating G-NAV requires uninstalling and reinstalling the application.

Uninstalling

Uninstalling the application can 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.

Storage requirements

The application only requires a minimum amount of storage space:

- Approximately 1MB for the WebAssembly module (execution module).
- No more than 30MB of data.

Privacy and security

In the current version, G-NAV does not transfer user information back to the web-server.

You will be asked for permission to switch on the geolocation module, which is a standard function of the browser.

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

FEATURES AND FUNCTIONS

General organization of the program

Screens

G-NAV is based on a main navigation screen where a column of 8 buttons provide access to functions and secondary screens. The content of each button is related to the state of the function.

The navigation between the screens has been designed to be as fluent as possible: there are no sub-screens and all screens other than the navigation screen provide a red NAV button on the top left corner to go immediately 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 page.

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. This aircraft is saved in the local storage and restored the next time the application is launched.

The general configuration of the aircraft is 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.

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-vector data is always displayed in green.

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 add a route, click on the green plus button. 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.

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.

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.

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.

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, 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 valid for 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.

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 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 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.

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.

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).

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

(N.A.)

Geographic data

Sectors

The moving map will display all loaded sectors using different colors:

- CTR
- TMA
- CTA
- RSA

References

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

- Wind turbines
- Cities
- Outlanding fields
- Woods
- Roads
- Railroads
- Rivers
- Lakes

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.

Automatic wind mode

NOTE: this function is under development and it has only been tested using a flight simulator.

When the glider maintains a steady turn rate, G-NAV estimates the wind based on the drift. This value can be adopted automatically when the AUTOMATIC 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.

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 (see *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.

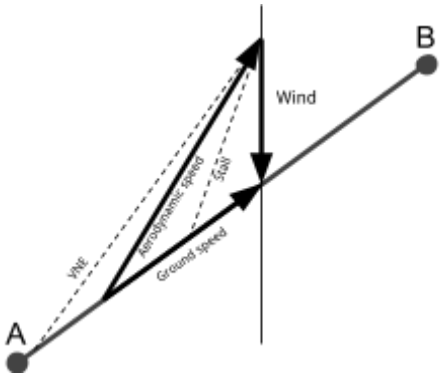
The equilibrium of forces for the provided weight at current altitude resolves into the gliding angle and the aerodynamic speed for each polar state. The gliding angle is given by the next equation,

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

and the aerodynamic speed by

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

The aerodynamic velocity 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.



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 sink rate.
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 depedent variables changes considerably.
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.

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 a long the route.

You are the only senor 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 air sink-rate

The expected air sink rate (ASR) is the average expected downward stream velocity of the air relative to the ground that one expects to find along the intended leg. 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 buttons that are always visible on the right-bottom corner of the moving map 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.

The evaluation of an appropriate ASR 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 finally be neutralized, 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 ASR 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 ASR.
- **Between -1.0 and -2.0: 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 ASR 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.

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 air sink-rate. The value is an estimate because it is affected by all inaccuracies inherent to the provided data (polar curve mismatching, wind and ASR 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.

Navigation 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

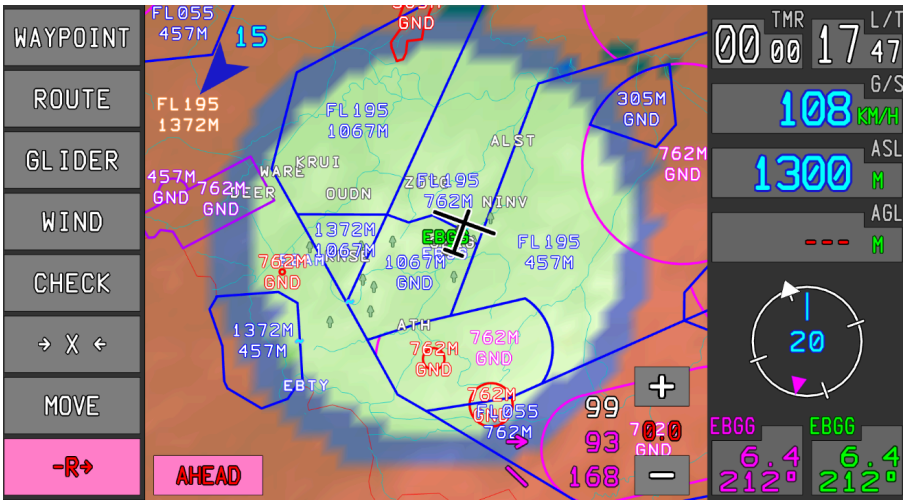
G-NAV uses the topographic data and the range estimation algorithms to represent the areas that are within maximum range in a straight line at optimal airspeed.

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.



Real time traffic

NOTE: this feature is under development.

G-NAV can be configured to poll for real time traffic data. It is important that the data is correctly timestamped, so that even when there is delay in the communication, G-NAV can adjust the representation correctly.

The position of the traffic is suggested as a wave front based on the last known position, track speed and time.