P2PHAPS – Java-based Balloon Trajectory Simulator V7

Martin Milnes, 20 Dec 2019

This document provides a summary of the P2PHAPS balloon trajectory simulator as so-far developed, for generating large numbers of simulated balloon trajectories using (archive) met. data for gridded wind speeds at multiple height levels over the North Atlantic, covering a range of temporal resolutions (eg monthly, daily and 6-hourly 2.5 degree gridded met. data) for all days in 2018. [Subsequently extended to support higher spatial &temporal resolutions, different years and multiple input met. data files]

The software is written in Java, and is based on the ImageJ (image processing) framework –which is used primarily for providing a simple (interactive) user interface for configuration setting, and for providing the background image on which wind vectors and/or tracks can be displayed.

Installation PreRequisites

- Java SDK [OpenJDK 13 (13.0.1), downloaded from https://jdk.java.net/13/ , or any other JDK or JRE should also work]
- ImageJ [http://wsr.imagej.net/distros/cross-platform/ij152.zip] – just unpack the zip file to a chosen location (eg D:/ImageJ; the top-level name will be ImageJ – this will be the ImageJInstallationDirectory. Then run the ImageJ.exe executable. You will also need to increase the memory allocated for ImageJ, by selecting Edit/Options/Memory&Threads and increase "Maximum Memory" to (eg) 24425 Mbytes.
- ActionBar (ImageJ plugin)
  [see https://imagejdocu.tudor.lu/doku.php?id=plugin:utilities:action_bar:start#installation]

Once the above have been installed:

- Create a directory structure for all the various files which will be needed for input and will be generated as output by the software, *for example*:

P2PHAPS = top-level directory (under <rootpath> wherever you choose to put that!)

Config = contains all config files
(in *P2PHAPSdefaults.txt* file, define:
PluginsDir <ImageJInstallationPath>/plugins/
ConfigDir <roothpath>/Config/
GeodesicFname <rootpath>/Config/Geodesic.csv,
LaunchSitesFname <rootpath>/Config/LaunchSites.txt,
LandingSitesFname <rootpath>/Config/LandingSites.txt,
GroundStationsFname <rootpath>/Config/GroundStations.txt)

Data = contains all met data files
(in *P2PHAPSdefaults.txt* file, define:
DataDir <rootpath>/Data
In WindsConfig* files, "FileUfname" and "FileVfname " are assumed to reside in the Data directory, eg:
"FileUfname ugrd_2019102100.bin",
"FileVfname vgrd_2019102100.bin")

Tracks = contains all Tracks outputs
(in *P2PHAPSdefaults.txt* file, define TracksDir as this directory,:
TracksDir <rootpath>/Tracks
Subdirectories will automatically be created under this TracksDir, with sequential run numbers auto-generated)

- Manually edit **ALL** configuration files which have hardwired pathnames; this has now been reduced to a single configuration file – P2PHAPSdefaults.txt [though these values can be over-ridden with run-specific files/directories in bespoke WindsConfig*.txt files, if required:

    - ***ImageJ/P2PHAPSdefaults.txt*:** This is the "master" pathname setting location – the file needs to be placed at the top level of the ImageJ installation directory (eg in D:/ImageJ); there are ten entries which need to be updated to reflect any specific local installation. The initial values are as shown below [*Note the extra "/" on the first two entries!*]:

```
PluginsDir D:/ImageJ/plugins/
ConfigDir D:/P2PHAPS/Config/
DataDir D:/P2PHAPS/Data
TracksDir D:/P2PHAPS/Tracks
SitesDir D:/P2PHAPS/Config
BackgroundImage D:/P2PHAPS/Config/LatLonNewNA.png
GeodesicFname D:/P2PHAPS/Config/Geodesic.csv
LaunchSitesFname D:/P2PHAPS/Config/LaunchSites.txt
LandingSitesFname D:/P2PHAPS/Config/LandingSites.txt
GroundStationsFname D:/P2PHAPS/Config/GroundStations.txt
DialogXpos 1140
DialogYpos 0
```

      The DialogXpos and DialogYpos settings determine the pixel position of the top left-hand corner of the main user dliag box – this may need to be modified, depending on each user's screen resolution (in pixels).

    - ***ImageJ/plugins/ActionBar/P2PHAPS.txt*:** Most functions will work without the need to edit this; only a couple of responses to button-clicks may need renaming of hardwired pathnames: "Connect Test" and "LaunchSites".

    - ***Data/WindsConfig*.txt*:** These have the capability to include hard-wired pathnames, but the new default is that any files (or subdirectory/file names) are under the directory "ConfigDir" defined in *P2PHAPSdefaults.txt*.

- Ensure all additional input files are in the locations pointed-to in the *P2PHAPSdefaults.txt*, *P2PHAPS.txt* and *WindsConfig*.txt* files (ie under Config, Data and Tracks subdirectories***; NB: only a subset of met data files are currently available in the shared GitHub repository***).
- Place the startup file *StartupMacros.txt* in the ImageJ/macros directory.
- Copy the (edited) *P2PHAPSdefaults.txt* file to the top-level ImageJ directory.
- Copy the *P2PHAPS.txt* file to the plugins/ActionBar directory.

  This contains a single executable line: run("Action Bar","/plugins/ActionBar/P2PHAPS.txt"); it refers to an ActionBar (defined by the script file: P2PHAPS.txt) which has been created to simplify the setting-up and running of MultiLevel_Vectors_Atlantic with a range of different input sets.

All CGI s/w is contained within a single java file: *MultiLevel_Vectors_Atlantic.java*.

A pre-compiled version of this is available: *MultiLevel_Vectors_Atlantic.class*

- If a new compilation is required, run:

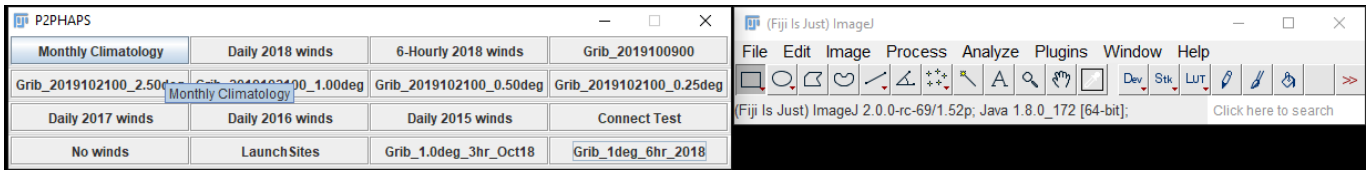
  javac –cp .;..\ImageJ.app\Contents\Java\ij.jar -Xlint:deprecation MultiLevel_Vectors_Atlantic.java

- If you have placed the source .java file somewhere other than <ImageJInstallaitonDir>/plugins then copy the compiled file *MultiLevel_Vectors_Atlantic.class* to the "plugins" subdirectory in the installed ImageJ directory (eg D:\ImageJ\plugins)

Starting the software

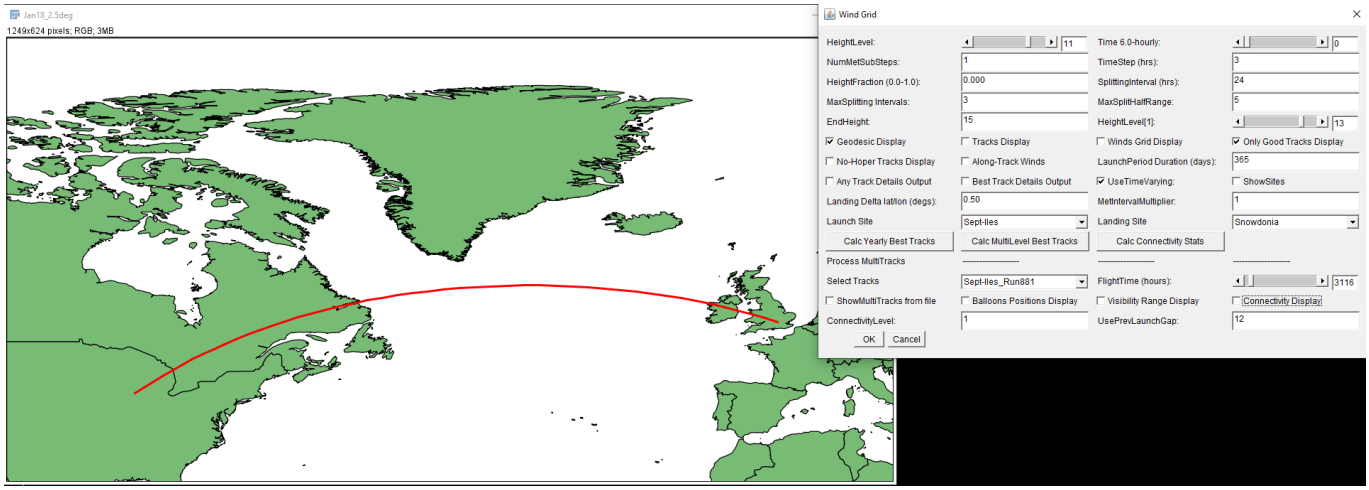
The use of the "AutoRun" and "ActionBar" facilities mean that the P2PHAPS software is automatically started when ImageJ is started. Thus, it is necessary to initiate ImageJ (eg through a desktop icon created when ImageJ was installed).

When ImageJ is started, the following windows appear:



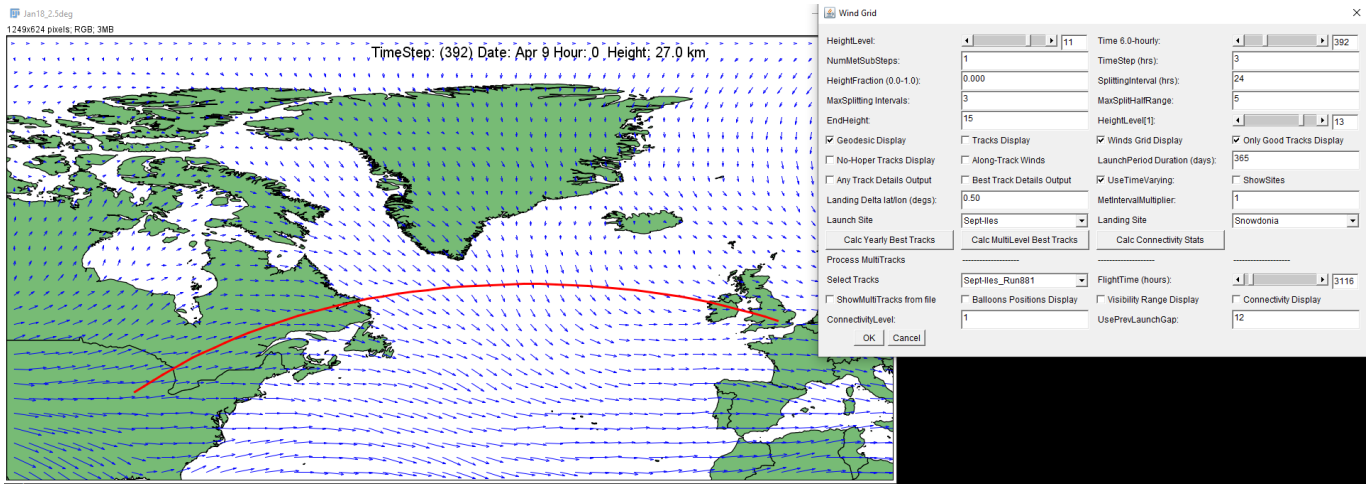
The right-hand window is the standard ImageJ menu, which is not needed for the current simulator.

To initiate the simulator, simply click on the button which identifies the preferred input met. data set (eg "Daily 2018 winds") which results in the following displays:



The right-hand menu controls the various setting for each run.

Whenever anything is changed in this GUI, the processing is recalculated, and results shown on the main map display, for example, setting a new height level:



To end a session, click on "OK" or "Cancel" in the right-hand "Wind Grid" control panel.

Most of the items in the "Wind Grid" control panel are self-explanatory, as outlined below:

Label	Description
Trajectory Modelling	
Height Level	Index to Height Level, in range 0 to 16, corresponding to the 17 standard met. data height levels (see below)
Time Day-of-year (0-364) Time QuarterDay-of-year Time 6-hourly Time 3-hourly Time Month (0-11)	Determines wind vectors displayed for specified Time/launch date. Interpretation and range of values depends on selected input met. data.
NumMetSubSteps	Normally=1, but can be set to >1 to allow multiple launches per Met Data interval (eg is 6-hourly met data, setting this value to 6 will result in hourly balloon launches)
Time Step (hrs)	Interval between updates in propagating the balloon flight. At each time step, the track is propagated by using the wind vectors (u,v) at the current (lat,lon,height) to determine a new position at the next TimeStep.
Height Fraction (0.0-1.0)	Allows specification of intermediate height level for single-height level run (or for final interval if multiple splitting intervals)
Splitting Interval (hrs)	Time intervals at which height level can be changed (when performing "MultiLevel Best Tracks" simulations). **NB:** Set to >200 to ensure no attempted splitting occurs!
Max No. Splitting Intervals	Maximum number of splitting intervals to apply (currently limited to 6). If the elapsed time for a flight exceeds SplittingInterval*MaxNoSplittingIntervals, then the height for all subsequent Splitting Intervals is set to the height assigned to the Splitting Interval corresponding to "MaxNoSplittingIntervals" (or 6, if MaxNoSplittingIntervals>6)
MaxSplitHalfRange	Default=5 ensures that all possible height levels (above 15kms) are considered at each splitting interval. Otherwise, a height-level jump at each splitting interval is constrained to current level +/- MaxSplitHalfRange levels.
EndHeight	Maximum Height Level for which trajectories are predicted (default=16 for 2.5 deg data, =14 for 1 deg data)
Height Level (1)	Provides ability to manually set the height level for the second splitting interval ("Height Level" at the top of the control panel sets "Height Level (0)" for the first Splitting Interval)
Geodesic Display	Toggle on/off display of Geodesic (between Chicago and London)
Tracks Display	Toggle on/off display of simple single-height-level balloon tracks (based on specified launch site and selected Height level and Time). Will follow displayed wind field if "UseTimeVarying" is unchecked (but not if that param is checked)
Winds Grid Display	Toggle on/off display of winds vectors (for Height Level and Day/Time indicated at the top of this window).
Only Good Tracks Display	Toggle on/off display of "good" balloon tracks when performing multiple simulations (ie one of "Best Tracks" check-boxes is set). Where "good" means landing location is within "Landing D delta lat Threshold" as set below)
No-Hoper Tracks Display	Toggle on/off display of all balloon tracks (even those with no prospect of reaching the target landing zone) – these trajectories are shown in cyan.
Along Track Winds	Toggle on/off display of wind vector at each Time Step
Launch Period Duration (days)	Time over which balloon launches are to be simulated when "Yearly Best Tracks" is selected

Any Track Details Output	Output per-time-step lat/lon/height values to csv file for all "good" tracks at each launch date/time (not just the "best" one for each launch date/time), where a "good" track is one which reaches the landing zone.
Best Track Details Output	Output per-time-step lat/lon/height values for the single "best" track from each launch date/time to MultiLevelBestTracks_<LaunchTime>.csv file (see below) (where "best" track is closest to geodesic over whole flight). If not checked then output will be a summary of all best tracks
UseTimeVarying	Toggle on/off use of time-varying met. data; if not checked then constant wind data is used throughout the balloon flight, if checked, then the met. data appropriate for the current time is used at each time step during the flight
Show Sites	Display all Launching and Landing Sites
Landing Delta lat/lon (degs)	Tolerance for balloon landing location being within specified separation from target landing location.
MetIntervalMultiplier	Default=1. Used to simulate reduced frequency of Met Data inputs (eg if input met data is 3-hourly, setting this value to 8 will simulate daily met data for the same time period).
Launch Site	Select Launch Site from list of LaunchSite files in *Sites* directory
Landing Site	Select Landing Site from list LandingSite files in *Sites* directory
Calc Yearly Best Tracks [button]	Finds best tracks (which any point in track which lies within Dlat Threshold distance of target landing site when reaching target longitude) for all days in the year (or all 6-hour intervals in year, *or all specified time intervals from Launch Start (defined by 2nd slider in window to N days later, where N is defined in the next setting below)*
Calc MultiLevel Best Tracks [button]	Supports using multiple splitting intervals to calculate tracks which get closest to target landing site
Calc Connectivity Stats [button]	Run connectivity statistics for each time snapshot through the year
Connectivity Modelling	***(All items below "Process MultiTracks")***
Select Tracks	Select multiple tracks/trajectories from subdirectories in *Tracks* directory
Flight Time (hours)	When "Plot Balloon Positions" is selected (see below), determines number of hours over which balloon flights are plotted
ShowMultiTracks from file	Toggle on/off ability to display multiple lat/lon tracks previously generated, by listing the csv files in a text file (MultiTracks.txt). Allows for visual comparison of tracks generated from different simulation runs – each track is shown in a different colour.
Balloons Positions Display	Plots positions of all balloons launched since Start Launch Date (determined by "Day-of-Year" in 2nd window slider above) at each time step since launch – as determined by "Balloon Flight Time" set by 3rd slider above.
Visibility Range Display	Show approximate line-of-sight visibility from each balloon/ground station (assuming simple minimum destination height of 15.8 kms)
Connectivity Display	Show levels of connectivity between balloons (and ground stations) at each time snapshot
Connectivity Level	Control level of connectivity for display
UsePrevLaunchGap	Maximum time interval between adjacent balloons for gap-filling

Generally, Connectivity modelling is based on a single file contains a list of "good" trajectories from multiple launches over a period of time from a single launch site. For example: "Sudbury_Run708_Yearly_Details_Met6hr_SubStep1_Split5_24.0h.csv".

However, it is possible to configure a "MultiTrack" file which contains a list of multiple such files (each with potentially different launch/landing sites). This is done by manually creating a new subdirectory under the Tracks directory (eg MultiTracks2), then creating a text file with the same name as the directory, but with ".txt" suffix, and editing this file to contain the list of trajectory files to be included, for example (*MultiTracks2.txt*):

```
D:\CGI\P2PHAPS\Tracks\Sudbury_Run708\Sudbury_Run708_Yearly_Details_Met6hr_SubStep1_Split5_24.0h.csv
D:\CGI\P2PHAPS\Tracks\Snowdonia_Run731\Snowdonia_Run731_Yearly_Details_Met6hr_SubStep1_Split4_24.0h.csv
```

When using such a file, all balloons from all input files will be assessed to determine which are in flight (and their location/altitude) at each snapshot in time during the modelled flight period.

Standard met. data height levels:

2.5 Degree (global) data			1.0deg (and higher resolution) data		
Index	**Pressure Level**	**Approx Altitude**	**Index**	**Pressure Level**	**Approx Altitude**
0	1000 hPa	0.1 kms			
1	925 hPa	0.8 kms			
2	850 hPa	1.5 kms			
3	700 hPa	3.0 kms			
4	600 hPa	4.2 kms			
5	500 hPa	5.6 kms			
6	400 hPa	7.2 kms			
7	300 hPa	9.1 kms	0	300 hPa	9.1 kms
8	250 hPa	10.4 kms	1	250 hPa	10.4 kms
9	200 hPa	11.8 kms	2	200 hPa	11.8 kms
10	150 hPa	13.5 kms	3	150 hPa	13.5 kms
11	100 hPa	15.8 kms	4	100 hPa	15.8 kms
12	70 hPa	17.7 kms	5	70 hPa	17.7 kms
13	50 hPa	19.3 kms	6	50 hPa	19.3 kms
14	30 hPa	21.6 kms	7	30 hPa	21.6 kms
15	20 hPa	23.3 kms	8	20 hPa	23.3 kms
16	10 hPa	25.9 kms	9	10 hPa	25.9 kms
			10	5 hPa	27 kms
			11	4 hPa	28 kms
			12	3 hPa	29 kms
			13	2 hPa	30 kms
			14	1 hPa	31 kms

In addition to the interactive user settings, a configuration file is used to define other values used by the software. Thirteen versions of this file currently exist, corresponding to using the monthly, daily or 6-hourly met. datasets:

- WindsConfig_MonthClimate.txt [2.5 deg, monthly, for whole year],
- WindsConfig_2018_Daily.txt [2.5 deg, daily, for whole year],
- WindsConfig_2018_Daily4x.txt [2.5 deg, 6-hourly, for whole year],
- WindsConfig_2017_Daily.txt [2.5 deg, daily, for whole year],
- WindsConfig_2016_Daily.txt [2.5 deg, daily, for whole year],
- WindsConfig_2015_Daily.txt [2.5 deg, daily, for whole year],
- WindsConfig_Grib_2019100900.txt [1.0 deg, 3-hourly for 3 days]
- WindsConfig_Grib_2019102100_1.00deg.txt [1.0 deg, 3-hourly for 3 days]

- WindsConfig_Grib_2019102100_0.50deg.txt [0.5 deg, 3-hourly for 3 days]
- WindsConfig_Grib_2019102100_0.25deg.txt [0.25 deg, 3-hourly for 3 days]
- WindsConfig_Grib_2019102100_2.50deg.txt [2.5 deg, 3-hourly for 3 days]
- WindsConfig_1.00deg_Oct18.txt [1.0 degs, 3-hourly for 3 months from 1st Oct 2018]
- WindsConfig_1.00deg6hr.txt [1.0 degs, 6-hourly for 12 months from 1st Jan 2018]

The original design supported a single pair of met. data files (one for U component and one for V component of wind vectors, identified in the WindsConfig file). However, as the system evolved, it became necessary to support multiple U and V files (typically one pair of files per month). This is managed through use of a *WindFilesList* file, which is identified in the required *WindsConfig* file.

The WindsConfig files are placed in the ImageJ/plugins directory. Using "*WindsConfig_2018_Daily.txt*" as an example, its contents are as shown below:

```
FileUfname ugrd_2019102100_4.bin
FileVfname vgrd_2019102100_4.bin
PlotGrid 1
arrowScale 1.0
headScale 0.15
MetNumX 120
MetNumY 60
MetCellSize 1.0
```

The first few lines identify the location and name of specified input files, covering:

FileUfname = File containing "uwinds" (ie East/west component of wind vectors)

FileVfname = File containing "vwinds" (ie North/south component of wind vectors)

WindFilesList = File containing list of multiple U-wind and V-wind files (supersedes FileUfname and FileVfname, if present)

Additional specific files can be specified (if the defaults defined in *P2PHAPSdefaults.txt* need to be changed):

GeodesicFname = File containing list of lat/lon co-ordinates for geodesic to be plotted on map.

LaunchSitesFname = File containing list of launch sites and their lat/long co-ordinates

LandingSitesFname = File containing list of landing sites and their lat/long co-ordinates

GroundStationsFname = File containing list of ground stations and their lat/long co-ordinates (must comprise two rows – first row is ground station for launch region, second is ground station for landing region)

Further (optional) configuration setting include:

PlotGrid = Interval for displaying gridded wind fields (if resolution finer than 1.0 degs, then need to set this greater than 1, so that only every N grid points are plotted)

PlotBalloonPos = Default setting for PlotBalloonPos in screen GUI – if set = 1 then no met. data will be loaded at startup (will speedup start time to initialise s/w, but means trajectory modelling can't be performed)

ShowMultiTracks = Default setting for ShowMultiTracks in screen GUI – if set = 1 then no met. data will be loaded at startup (will speedup start time to initialise s/w, but means trajectory modelling can't be performed)

arrowScale = Controls scaling of displayed arrow lengths in relation to wind speed

headScale = Controls size of displayed arrowheads

TargetEndLat = Default latitude for landing location

TargetEndLon = Default longitude for landing location

MetNumX = Number of X-position (columns) in binary file (for Grib2-derived data)

MetNumY = Number of Y-position (rows) in binary file (for Grib2-derived data)

MetCellSize = Spatial resolution of met data (in degrees, eg 1.0, 2.0, 0.25 etc)

MetTimeStep = Temporal resolution of met data (in hours)

MaxSplitIntervals = Number of splitting intervals to apply for trajectory modelling

BalloonLaunchTime = Default setting for BalloonLaunchTime in screen GUI

SelectedMultiTrackFile = Default setting for "Select Tracks" in screen GUI [NB: This is an integer index to the selected value from the "current" drop-down menu; would change if new options are added]

SelectedLaunchSite = Default setting for "Launch Site" in screen GUI [NB: This is an integer index to the selected value from the "current" drop-down menu; would change if new options are added]

SelectedLandingSite = Default setting for "Landing Site" in screen GUI [NB: This is an integer index to the selected value from the "current" drop-down menu; would change if new options are added]

DialogXpos = Screen X-position (in pixels) for top left corner of main dialog box (floating in centre, if not specified)

DialogYpos = Screen Y-position (in pixels) for top left corner of main dialog box

Additional Input Configuration Files

Met Data Files

a) 2.5 degree global met data (monthly/daily/6-hourly)
Pairs of binary files comprised of 32-bit float values (one for U-component [East/West], one for V-component [North/South] of wind vector). Each file contains global data.
Ordered as:

```
For each pixel row
    For each met data time step
        For each height level
            For each pixel column
                Increment index to data
```

b) 1.0 degree (and finer spatial resolution) met data (3-hourly/6-hourly)
Same format as (a) but covering only lon/lat limits: -105 degs to +15 degs lon, 30 degs to 90 degs lat.
Ordered as:

```
For each height level
    For each met data time step
        For each pixel row
            For each pixel column
                Increment index to data
```

c) WindFiles List

Comprises a list of actual Met data files (U-component, V-component). The identified *WindFilesList* file must contain an ordered sequence of wind files in the sequence:

```
U-windfile1
V-windfile1
```

```
U-windfile2
V-windfile2
:
U-windfileN
V-windfileN
```

Where each windfile is specified by its full path and filename. This file needs to be manually created, as required.

File(s) containing lat/lon pairs for geodesic paths

One line per location, each line contains <latitude>,<longitude> in degrees (eg *Geodesic.csv*).

Sites Locations Files

Site Locations files all have a similar format, comprising one or more rows, each consisting of:
"<name> <latitude> <longitude>" (in degrees)

a) Launch Sites List & per-LaunchSite files

b) Landing Sites List & per-LandingSite files

c) Ground Station Sites file
This file requires two rows; the first row corresponds to the ground station near the launch site, and the second row corresponds to the ground station near the landing site.

For single-site files, just a single row is included in the file.

Trajectories Files *(output from Trajectory Modelling, used as input to Connectivity Modelling)*

MultiTracks File List & per-MultiTrack files; format is as per the "Details" files output by the trajectory modelling.

Batch Scripts

All the processing is based on Java plugins running under the ImageJ environment. This also supports scripting which can be used to define specific configuration settings based on any of the values which are normally set using the interactive GUI. Furthermore, ImageJ itself can be called from the command line, avoiding the need for any user interface. This can be useful for running multiple scenarios over a range of different conditions. For example:

```
java -jar ij.jar CommandLine.txt
```

An example invocation of “MultiLevel Vectors Atlantic" from a script file is:

```
run("MultiLevel Vectors Atlantic", "heightlevel=11 time=0 flighttime=0
nummetsubsteps=1 timestep=3 heightfraction=0.000 splittinginterval=24 maxsplitting=1
maxsplithalfrange=5 heightlevel[1]=13 geodesic no-hoper along-track launchperiod=365
usetimevarying landingthreshold=0.50 metintervalmultiplier=1");
```

An example script looping over multiple scenarios is:

```
/ Script to perform same run conditions over multiple launch sites

NumMetSubSteps=1;

NumSplits=3;

SplitInterval=24;

LaunchStart=0;

LaunchPeriod=365;

// Load background and ensure correct Met Data configured

open("D:/CGI/P2PHAPS/ImageJ/LatLonNewNA.png");

rename("6-Hourly 2018 winds");

File.copy("D:/ImageJ/plugins/WindsConfig_2018_Daily4x.txt","D:/ImageJ/plugins/WindsConfig_Default.txt");

Sites = newArray("Aurora",”Cartwright,”Sept-Iles”,”Sudbury”)

for (i=0;  i<Sites.length;  i++)

{

   Name = Sites[i];

   // Copy LaunchSites_<name>.txt to LaunchSites.txt

   File.copy("D:/CGI/P2PHAPS/ImageJ/LaunchSites_"+name+".txt", "D:/CGI/P2PHAPS/ImageJ/LaunchSites.txt");



   val1="time="+LaunchStart;

   val2=" nummetsubsteps="+NumMetSubSteps+" timestep=3 splittinginterval="+SplitInterval+"
maxsplitting="+NumSplits;

   val3=" maxsplithalfrange=5 geodesic only_good yearly launchperiod="+LaunchPeriod+" best_track
usetimevarying";



   // Run MultiLevel_Vectors_Atlantic with specified args

   run("MultiLevel Vectors Atlantic", val1+val2+val3);

}
```

Example Output files:

<LaunchSite>_Run<RunNumber>_BestTracksNoDetail_<LaunchTime>_Met<interval>.csv

```
NumIntervals,LaunchTime,H0,H1,H2,MinDlat,EndTime
3,238,11,11,12,0,136.34
3,238,11,12,13,0.24,166.49
3,238,12,11,13,0.01,159.87
```

<LaunchSite>_Run<RunNumber>_BestTracks_<LaunchTime>_Met<interval>.csv

```
LaunchTime,TrackNum,StepTime,Latitude,Longitude,Height,GeodesicRMSE
  231,  0,   0.0, 53.88, -56.328,  11.000, 525.839
  231,  0,   3.0, 54.05, -55.653,  11.000, 525.839
  231,  0,   6.0, 54.23, -54.975,  11.000, 525.839
:
  231,  0, 231.0, 53.79,   2.883,  11.000, 525.839
  231,  0, 234.0, 53.90,   3.934,  11.000, 525.839
  231,  0, 237.0, 54.01,   4.988,  11.000, 525.839
```

<LaunchSite>_Run<RunNumber>_Yearly_Summary_Met<interval>_SubStep<n>_Split<m>_<interval>h.csv

Same format as *BestTracksNoDetail,* but with multiple Launch times.

<LaunchSite>_Run<RunNumber>_Yearly_Details_Met<interval>hr_SubStep<n>_Split<m>_<interval>h.csv

Same format as *BestTracks,* but with multiple Launch times.

<LaunchSite>_Run<RunNumber>_Summary_Connectivity<Level>Stats.csv

```
Hour  PathLength  NumberNodes
0     0     2
3     0     2
6     0     2
:
```

<LaunchSite>_Run<RunNumber>_Details_Connectivity<level>Stats.csv

```
Hour  BallonNum   LaunchTime  Lat   Lon         Height      GeodesicRMSE
105   0           0           53    -3.5        0           525.839
105   1           30          52.33 -11.214     17.794      299.697
105   2           42          53.22 -15.415     15.904      1040.905
105   3           60          52.09 -30.747     19.854      1070.597
:
```

<LaunchSite>_Run<RunNumber>_NewTracks.csv

Hour	BallonNum	LaunchTime	Lat	Lon	Height	GeodesicRMSE
105	0	0	53	-3.5	0	525.839
105	1	30	52.33	-11.214	17.794	299.697
105	2	42	53.22	-15.415	15.904	1040.905
105	3	60	52.09	-30.747	19.854	1070.597
:						

<LaunchSite>_Run<RunNumber>_LonBins_<level>Stats.csv

Time	Type	70W	60W	50W	40W	30W	20W	10W	0W
0	0	0	2	1	1	0	1	2	0
0	1	5	20	14	14	14	14	17	5
0	2	32	80	75	64	60	66	68	10
0	3	945	2334	2089	1695	1647	1586	1517	494
1	0	0	2	1	1	1	1	1	0
1	1	4	12	10	9	9	8	8	3
1	2	10	33	29	26	27	21	21	11
1	3	867	2356	2234	1568	1684	1474	1395	437
:									

Software Structure

Processing Structure:

Trajectory Modelling

Key steps (for a single launch time):

For splitting interval 1:

For each height level in a specified range:

For splitting interval 2:

For each height level in a specified range:

:

For splitting interval N (upto a specified number):

For each height level in a specified range:

For each track propagation timestep (within splitting interval):

- get (lat/lon/height) from previous timestep and met-data for current time
- propagate lat/lon to next timestep (eg every 1-hour)
- if longitude or time beyond specified limits then terminate track

Once a track has terminated, check suitability of track in terms of closeness to landing zone:

- If time exceeded limit or latitude difference excessive, then flag as “no-hoper” (ie never reaches required longitude), and flag as “cyan track”
- If latitude too far from landing site latitude then flag as “blue track”
- If final height level gives final Lat North of landing site and final height level +/- 1 gives final Lat South of landing site (or vice versa), then flag as “black track”
- For each black track, iterate (non-linear) solution based on intermediate height levels to get within specified tolerance for landing site lat/lon
- For all black tracks with converged solutions, select “closest” to geodesic (in terms of RMSE distance of each timestep location) as “best” and flag as “yellow track”

Connectivity Modelling

*This is still under development, and should **not** be relied-upon for making decisions regarding overall connectivity.*

Software Modules

Initialisation:

```
run();
showDialog();
    ReadConfigFile();
    ReadMultipleWindFiles();
        ReadGribWinds();
        ReadWinds();
ListSites();
ListSites();
ListFiles();
```

Operations:

```
dialogItemChanged();
        MonthFromDoY();
        ReadLaunchSitesFile();
        ReadLaunchSitesFile();
        ReadGroundStationsFile();
        ReadLLfile();
        PlotGridField();
            DrawArrow();
        DrawLLseq();
        PlotSites();
        DrawMultiTracks();
            ReadTrackFile();
        PlotTracks();
            DrawArrow();
        LoadBestTracks();
        MonthFromDoY();
        CalcPaths();
            PlotLocation();
            OrderConnections();
            CalcBestTracksForAllSplitIntervals();
```

```
                CalcConnectivity();
                        CheckVisibility();
                        GeodesicDistance();
        ListDirs();
        OutputRunConfigInfo();
        CalcBestTracksForAllSites();
                PlotTracks();
        ListDirs();
        OutputRunConfigInfo();
        CalcBestTracksForAllSplitIntervals();
                CalcBestTracksForAllSites();
                PlotTracks();
actionPerformed();
        runYearly();
                ListDirs();
                OutputRunConfigInfo();
                CalcBestTracksForAllSplitIntervals();
        runMutiLevel();

        CalcConnectivityStats();
                CalcPaths();
```

Functions: (class MultiLevel_Vectors_Atlantic)

Main Control (based on standard ImageJ framework)

public void run(ImageProcessor ip)

Main control function – entry point for ImageJ processes.

public int showDialog(ImagePlus imp, String command, PlugInFilterRunner pfr)

Primary User Interface, providing ability to set/modify configuration parameters and to initiate scenario simulations; all parameters can also be specified in a command line for batch processing from a script (in which case this GUI is not displayed).

public boolean dialogItemChanged(GenericDialog gd, AWTEvent e)

Callback function called in response to any item being changed in the main GUI dialog box.

public void actionPerformed(ActionEvent e)

Callback function called in response to any button being clicked in the main GUI dialog box – one of: “Calc Yearly Best Tracks”, “Calc MultiLevel Best Tracks”, “Calc Connectivity Stats”

Utility Functions for Trajectory Modelling

void ListDirs(String topdir)

Lists all directories in the “Tracks” directory, to find the highest previous run number; used to determine next (incremented) run number for the next simulation run.

void ListFiles(String topdir, String wildcard)

Lists all files in specified directory matching specified wildcard – used to identify all “Yearly_Details” files, to display in dropdown picklist for “multitracks” to be used as input to connectivity processing.

void ListSites(String topdir, String wildcard)

Lists all files in specified directory matching specified wildcard – used to identify all “LaunchingSites” and “LandingSites” files, to display in dropdown picklists for selection of specific launch/landing site to use in a trajectory simulation run.

int MonthFromDoY(int iDoY)

Simple conversion from DoY to month number.

File Readers for Trajectory Modelling

void ReadConfigFile()

Reads the WindsConfig_Default.txt file from the ImageJ/plugins directory, to obtain configuration parameter values for subsequent processing/display.

void ReadLaunchSitesFile(String fname)

Reads the LaunchSites_<opt>.txt file based on the selected launch site, to obtain lat/lon values for subsequent processing.

void ReadLandingSitesFile(String fname)

Reads the LandingSites_<opt>.txt file based on the selected landing site, to obtain lat/lon values for subsequent processing.

void ReadGroundStationsFile(String fname)

Reads the GroundStations_<opt>.txt file based on the selected landing site, to obtain lat/lon values for subsequent processing.

void ReadWinds()

Reads the (global) 2.5-deg winds met. data (monthly, daily, or 6-hourly) and extracts those values within the min/max lat/lon limits used for the scenario modelling (-105 to +30 degs longitude, 30 to 90 degs latitude).

void ReadGribWinds()

Reads the 1-deg winds met. data (6-hourly or 3 hourly; also 0.5deg, 0.25deg) from files which have already been limited to be within the min/max lat/lon limits used for the scenario modelling (-105 to +30 degs longitude, 30 to 90 degs latitude).

void ReadMultipleWindFiles()

Reads a text file selected from the "Select Tracks" dropdown menu, which contains a list of multiple "Yearly_Details" files, each containing multiple trajectories from one launch site.

void ReadLLfile(String fname)

Reads a text file containing a sequence of lat/lon pairs (eg for geodesic plotting).

void ExtrpolateHeights( )

Test function to explore adding a linearly-extrapolated height level above the highest one read from the input met data files.

void CalcLonDistribution(double lon, double lat, Color colTrack)

Test function to explore the longitudinal distribution of balloons at each timestep for different categories of track (based on displayed track colour – cyan/blue/black/yellow)

Core Processing for Trajectory Modelling

void runYearly(AtomicInteger progress)

Main control function for looping over all launch times from specified start time over specified period at specified intervals (all determined by options selected in main GUI). Initiated by clicking on "Calc Yearly" button.

void runMultiLevel()

Main control function for single launch time trajectory modelling; Loads up run configuration data, then runs CalcBestTracksForAllSplitIntervals() – see below.

void CalcBestTracksForAllSplitIntervals(PrintWriter writer,int imTimeStep, double SplittingInterval, int StartLaunchTime, int MaxLaunchTimes)

For specified range of launch times, loops over a range of splitting intervals (from none to the specified (maximum) number of splitting intervals, using CalcBestTracksForAllSites() – see below.

void CalcBestTracksForAllSites(PrintWriter writer, int imTimeStep, int SplitInterval)

For specified site(s) and splitting intervals, at each splitting interval, loops over all specified heights, (+/- N from "current" height [ie height from previous splitting interval], propagating a trajectory forward in time at the new height level, using PlotTracks() – see below.

void PlotTracks(PrintWriter writer, Color colTrack, boolean ShowTracks, int site, int imTimeStep, double StartLat, double StartLon, double timestep, int interp, int iheight, double frac, int ApplyFracInterval)

This is the core trajectory propagation function – it contains the most important part of the processing in terms of determining the balloon trajectories based on the met. data and splitting intervals. It also includes the checks for determining how good each trajectory is, in terms of reaching the target landing zone.

void TimeHeightInterpolation(double latind, double lonind, int dataTimeStep, double deltaTime, int TrackHeight, double heightfrac)

At each trajectory propagation timestep, the met data to use is interpolated in time and in height, to obtain a linear-interpolated (u,v) value (from the input met. data gridded values) for the "current" time and altitude.

Plotting/Output Functions for Trajectory Modelling

void DrawArrow(double x1, double y1, double x2, double y2, Color col)

Draws an arrow between specified (x,y) pixel locations, with characterisitcs as defined in the Winds_Config file.

void DrawLLseq(int Num, Color col, int Thickness)

Draws a connected line through sequence of lat/lon pairs (as read using ReadLLfile()).

void PlotSites()

Shows locations and names of potential launch sites, landing sites and ground stations (as determined from reading the corresponding text files with the names and locations of each)

void PlotGridField(int iheight, int imTimeStep, Color col)

Displays wind field as vectors (from selected met. data files) for selected date/time and height, with arrows scale and spatial subsampling (if needed, eg for higher resolution met. data) defined in WindsConfig file.

void OutputRunConfigInfo(String NewRunName)

Outputs <LaunchSite>_Run<RunNum>_Config.txt file for current simulation run.

Connectivity Processing Functions *[NB: Currently still under development]*

Utility Functions for Connectivity Modelling

double GeodesicDistance(float Lat1, float Lon1, float Lat2, float Lon2)

Calculates the shortest path between two specified (lat/lon) locations at the Earth's surface.

double VisibilityRange(double BalloonHeight, double TargetHeight)

Calculates the maximum range (in kms) where a line-of-sight is possible between two objects at two different heights above the Earth's surface.

boolean CheckVisibility(int ind1, int ind2)

Uses "VisibilityRange()" to determine whether balloon indexed by ind2 is visible from balloon indexed by ind1

void RemoveBalloons(int from, int to)

Remove all balloons in index range "from" to "to" inclusive.

void RemoveShortLinks(int ind0)

Remove all balloons which are closer than the furthest at which visibility is still possible.

File Readers for Connectivity Modelling

void ReadTrackFile(String rowList, int YearlyTracks, Color col)

Reads "multitrack" file specified by "Select Tracks" dropdown menu, and plots all trajectories from the file, in the specified colour.

void ReadTrackBalloons(String rowList)

Reads specified csv file, populating balloons arrays with lat/lon/height for each timestep in each trajectory

void LoadBestTracks()

If selected MultiTrack file contains "MultiTrack" then loops over all csv files in MultiTrack file (calling ReadTrackBalloons for each csv file), otherwise calls ReadTrackBalloons for single csv file.

Core Processing for Connectivity Modelling

void CalcConnectivityStats()

For all specified time steps, calculate the connectivity paths for all balloons currently in-flight at that time (using CalcPaths()), and output results in "Summary_Connectivity" and "Details_Connectivity" output csv files. Also, for newly generated trajectorirs, used for filling-gaps, output a "NewTracks" csv file.

void CalcConnectivity(int iTime, PrintWriter connectwriter)

For a given set of balloons (in longitude order), calculate the overall connectivity along the track which joins them all, and the associated path length (and RMSE from geodesic).

void OrderConnections()

Sorts all balloons for current timestep into longitude order

void InsertNewConnectionLocation(int ind0, float NewLaunchTime, float NewLat, float NewLon, float NewHeight)

Inserts a new balloon/connection at the specified index, with the specified new launch time, lat,lon & height [*initially based on simple linear interpolation of values either side*]

void CalcPaths(int iTime, PrintWriter connectwriter, PrintWriter newtrackswriter)

Main Connectivity processing function; for the current time, determine the positions of all balloons currently in-flight, and determine visibility between successive balloon; if a further balloon has visibility, the remove this one from the list of balloons needed for connectivity; if the gap between successive balloons is too great for line-of-sight visibility then fill the gap by simulating an intermediate launch time,

Plotting/Output Functions for Connectivity Modelling

void PlotLocation(float Lat, float Lon, float Height, float Height2, float fLaunch, Color colour)

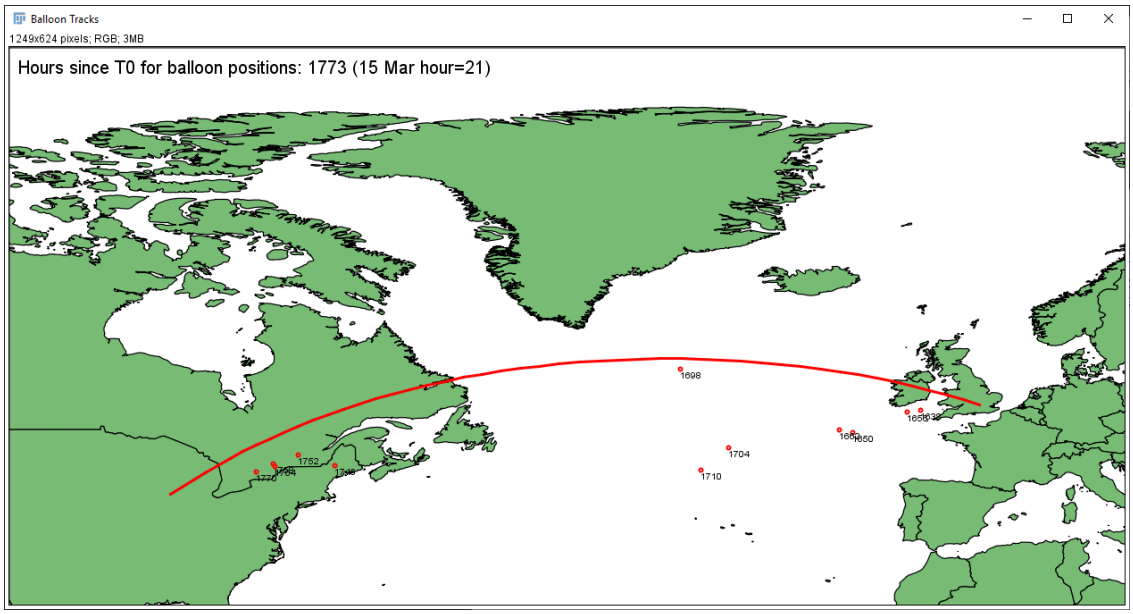
Shows the position of a balloon with associated launch time identifier, optionally with a "visibility ellipse" showing the extent of line-of-sight range for another balloon at *Height2* compared to current balloon at *Height*.

void DrawMultiTracks(String TrackListFname)

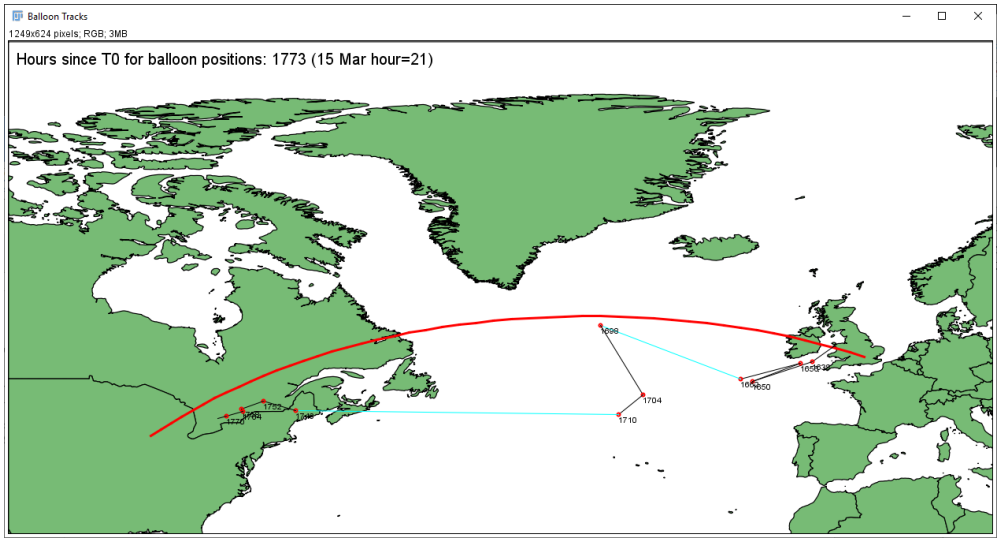
If selected MultiTrack file contains "MultiTrack" then loops over all csv files in MultiTrack file (calling ReadTrackFile for each csv file), otherwise calls ReadTrackFile for single csv file.

Example Connectivity Plots

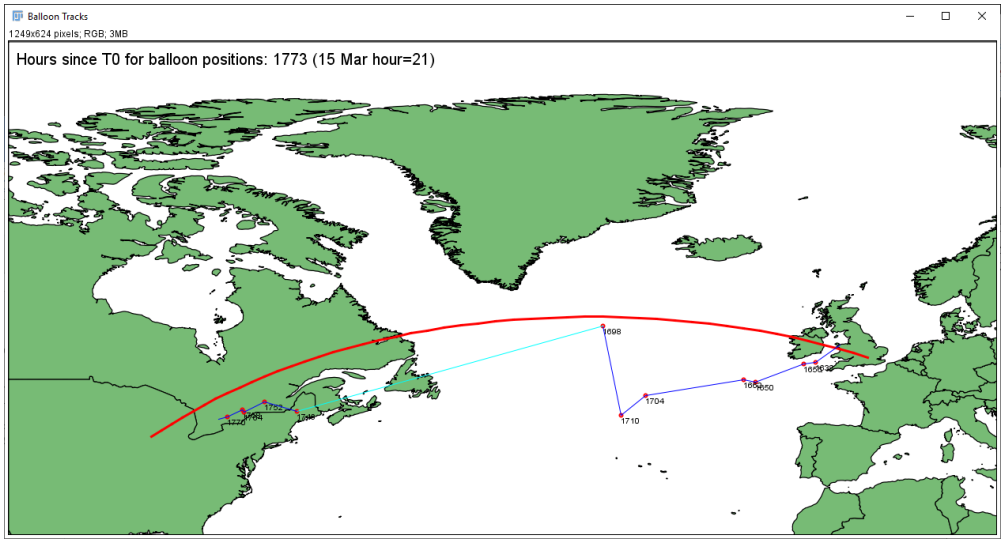
Balloon Positions:



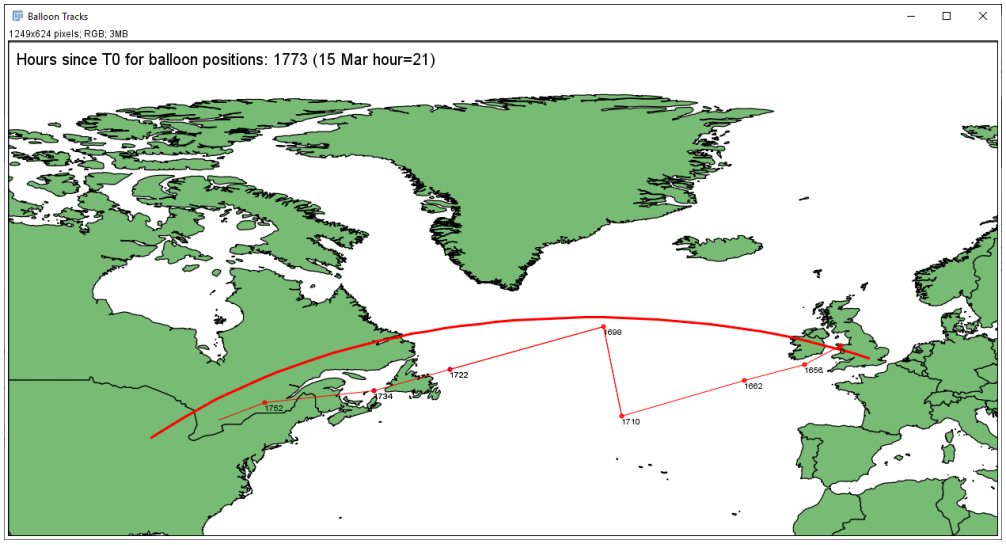
Initial Connectivity - Launch order [Connectivity Level = 1]:



Longitude-ordered [Connectivity Level = 2]:

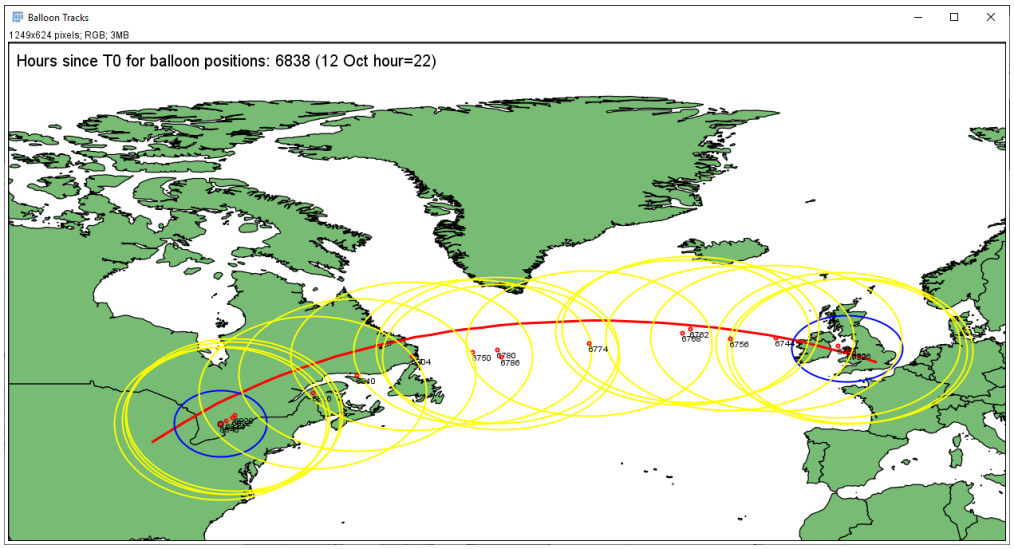


First iteration of gap-filling (binary splitting, based on launch times)

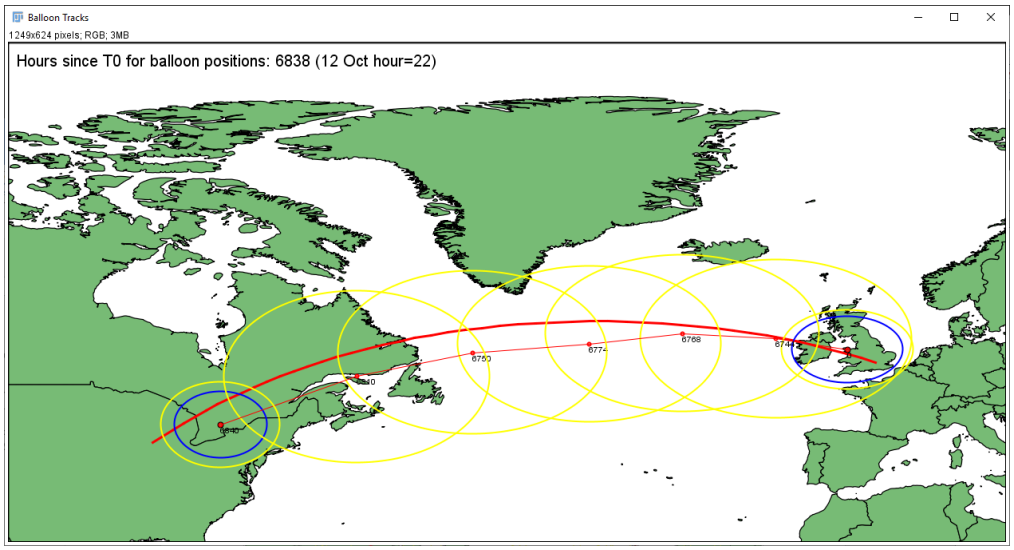


Line-of-Sight visibility ranges:

Original (all balloons within time snapshot):

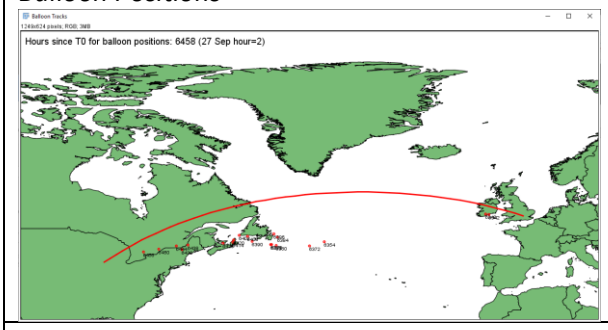


After "optimal" connectivity:

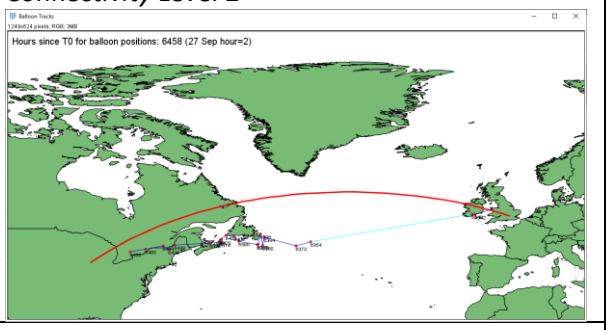


Example of iterative binary splitting for higher connectivity processing "levels"

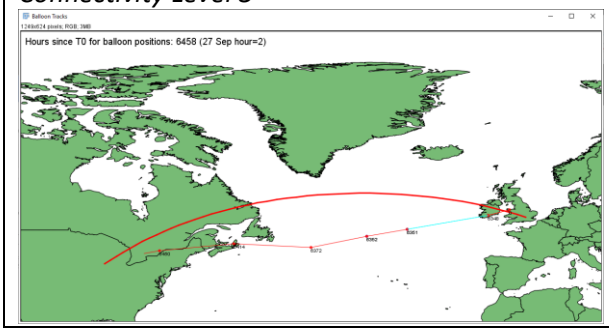
Balloon Positions



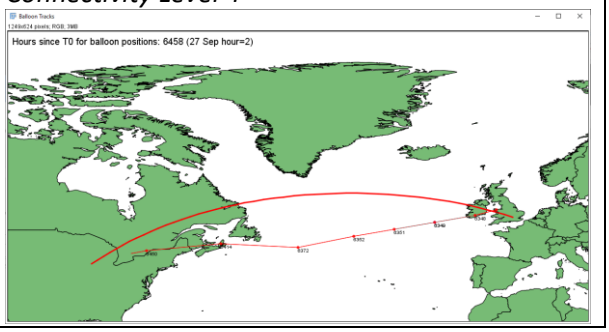
Connectivity Level 2

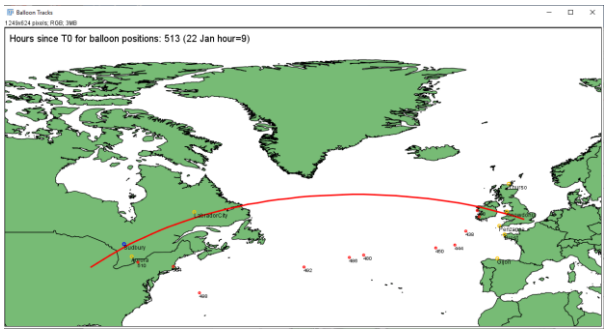


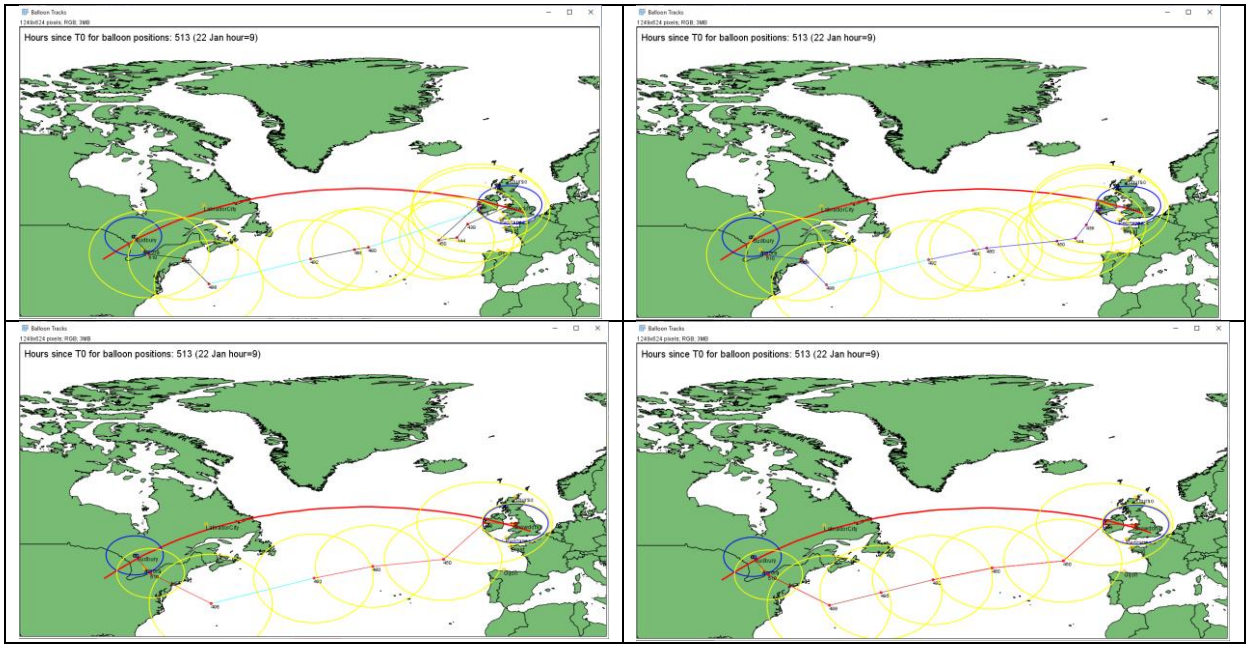
Connectivity Level 3



Connectivity Level 4

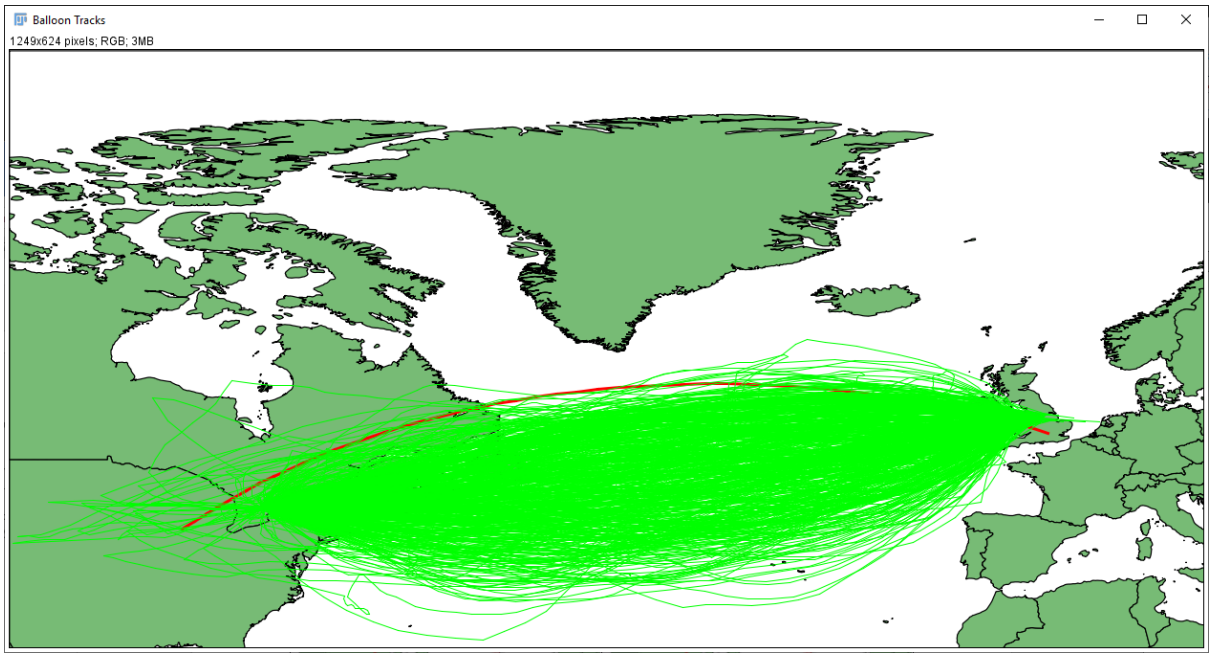






Show MultiTracks from File

Example of plotting all “good” tracks (for one year) from one launch site to one landing site (Aurora to Snowdonia):



Example of two sets of tracks (West to East and East to West), with overlay of balloon positions for selected date/time.