CIVL 2020 PLENARY – ANNEXE 30A
SOFTWARE WORKING GROUP PROPOSAL — CHANGES TO SECTION 7F

Contents

1 Overview of changes

1. Remove final glide decelerators
2. No more prescribed turnpoint direction (including start)
3. Goals will always be cylinders from 2021
4. Clarify task distance calculation
5. Clarify rule for restarts for races with multiple start gates and for elapsed time tasks
6. Use a constant leading weight for paragliding
7. Task results are given with one decimal point, only round once when calculating competition results
8. Adopt the PWCA's leading points calculation for paragliding
9. Adopt the PWCA's time points calculation for paragliding
10. No more minimum time for stopped tasks
11. Stopped tasks: Redistribute removed time points as distance points
12. FTV: Use best score for FTV validity

2 Explanations

2.1 Remove final glide decelerators

Final glide decelerators have not been in use for several years, and it seems very unlikely they will ever be used again, so we remove them from the rules.

2.2 No more prescribed turnpoint direction (including start)

The indication whether a turnpoint is "enter" or "exit" is usually implicitly given through the task, and is not enforced by the current implementation in FS. By removing the indication, we remove a discrepancy between rules and implementation. We also remove a potential source for task setting errors (a turnpoint is declared "exit" when in fact the software will implicitly treat it as "enter").

The main implication of this is with the start, where FS until now checks that the start line is crossed in implicitly prescribed direction. Removing this requirement allows for more alternative start scenarios especially when large cylinders are being used.

2.3 Goals will always be cylinders from 2021

With the use of very big cylinders, the direction of the goal line, which is derived from a cylinder center, becomes ambiguous. Since in most of the tasks, an ESS away from goal is being used nowadays, a goal line is no longer necessary, and can easily be replaced by a goal cylinder of adequate size. This allows pilots to fly into goal from any direction.

To prevent a situation where task setters who are unaware of this change set a task with a goal line, which then must be changed after the fact to score the task, we introduce this change over 2 years. In 2020, we still allow the use of the goal line, but educate task setters and score keepers to the effect that they stop using it. From 2021, the only goal shape will be the cylinder.

2.4 Clarify task distance calculation

Task distance calculation has so far been insufficiently documented, with some ambiguity for example regarding the start of the measurement. We rectify this in 2020.

2.5 Clarify rule for restarts for races with multiple start gates and for elapsed time tasks

The rule for restarts in tasks where this is possible (races with multiple start gates and elapsed time tasks) require clarification.

2.6 Use a constant leading weight for paragliding

In paragliding, the PWCA's maximum leading weight of 16.2% is now uniformly applied to all situations, regardless of goal ratio.

2.7 Task results are given with one decimal point, only round once when calculating competition results

To avoid compound rounding effects, task results will be published with one decimal, and rounding will only occur once, when calculating the total competition score.

2.8 Adopt the PWCA's leading points calculation for paragliding

The PWCA's leading points calculation, which weighs the leading coefficient across the speed section distance, to reduce the importance of leading out very early in task, was already applied to the 2019 Paragliding World Championships, and will now be officially adopted by CIVL.

2.9 Adopt the PWCA's time points calculation for paragliding

The PWCA's time points calculation, with a less pronounced points degradation between the first pilots in goal, was already applied in the 2019 Paragliding World Championships, and will now be officially adopted by CIVL.

2.10 No more minimum time for stopped tasks

Stopped tasks are valid tasks, regardless of their duration. Their sportive merit depends on their validity, including the stop validity. This makes the minimum time requirements currently in place superfluous. Through the hard threshold of these time requirements, the time at which a task stop is announced can make a huge difference in task and competition outcome. Without the time requirements, meet directors will never be tempted to run a task a bit longer, to make it valid. They can stop it at any time they feel a task needs to be stopped, without fear of losing a valid task.

2.11 Stopped tasks: Redistribute removed time points as distance points

In stopped tasks with pilots in goal, their time points are reduced by a certain amount, to remove a discontinuity between pilots just before or after ESS at task stop time. To date this leads to an unwanted task devaluation, since the sum of the distributed points will be less than what should be distributed according to the day validity.

We rectify this by adding the removed time points to the available distance points.

2.12 FTV: Use best score for FTV validity

For FTV to properly work in all cases, pilot performances must be compared with the winner's performance, rather than the theoretical points maximum in a task. This is how FTV has been calculated in the PWCA for several years. This was already applied in the 2019 Paragliding World Championships, and will now be officially adopted by CIVL.

3 Changes in Details

3.1 4.2.1 Effect of altitude in distance calculation

3.1.1 Status quo

In paragliding, for final glide decelerators (5.6) and altitude bonus in stopped tasks (12.3.6), altitude is also considered, but this does not affect distance calculations between two geographic points.

In hang gliding, for altitude bonus in stopped tasks (12.3.6), altitude is also considered, but this does not affect distance calculations between two geographic points.

3.1.2 Proposal

For altitude bonus in stopped tasks (12.3.6), altitude is also considered, but this does not affect distance calculations between two geographic points.

3.1.3 Reasoning

Removed final glide decelerators, no more distinction between PG and HG necessary.

3.2 4.2.2 Distance calculation on the FAI sphere

3.2.1 Status quo

The distance between two points on the FAI sphere, identified by their radian coordinates lat1/long1 and lat2/long2, is calculated using the Haversine formula.

$$distLat = lat2 - lat1$$

$$distLong = long2 - long1$$

$$a = (\sin\frac{distLat}{2})^2 + \cos lat1 * \cos lat2 * (\sin\frac{distLong}{2})^2$$

$$radianDistance = 2 * \arctan 2(\sqrt{a}, \sqrt{1-a})$$

$$distance = radianDistance * 6371000$$

To reproduce this formula in Excel, the following modification is necessary due to a different implementation of the arctan2 function:

$$radianDistance = \pi - 2 * \arctan 2(\sqrt{a}, \sqrt{1-a})$$

3.2.2 Proposal

-

3.2.3 Reasoning

All distance calculation is now done on WGS84

3.3 5. Competition Parameters

3.3.1 Status quo

In paragliding competitions, the following must also be defined by the same person or group who defines the first five competition parameters above:

6. Final glide decelerator
7. Score-back time for stopped task

3.3.2 Proposal

In paragliding competitions, the following must also be defined by the same person or group who defines the first five competition parameters above:

6. Score-back time for stopped task

3.3.3 Reasoning

No more final glide decelerators

3.4 5.6 Final Glide Decelerator

3.4.1 Status quo

The concept of a final glide decelerator was introduced to counteract a development in competition paraglider design which favoured stability at high speeds over stability at trim speed. The two final glide decelerators available are:

- Conical end of speed section (CESS): Instead of a cylinder, the end of speed section is an inverted cone. Time stops for a pilot when they enter that cone. For details see 6.2.2.
- Arrival altitude time bonus (AATB): The time bonus is calculated based on each pilot's altitude above goal when crossing the end of speed section cylinder. This bonus is then deducted from the pilot's speed section time to determine the score time. See also 8.6.
-

A meet director may choose to use no final glide decelerator, or use either of the two outlined above.

3.4.2 Proposal

-

3.4.3 Reasoning

No more final glide decelerators

3.5 6.1.1 Race task

3.5.1 Status quo

A race task definition consists of:

1. A launch point, given as GPS coordinates
2. A number of control zones
3. A goal
4. An indication which of the control zones is the start (start of speed section)
5. If goal does not serve as end of speed section: An indication which of the control zones is the end of speed section, along with its specific parameters (such as incline and radius for CESS, altitude time bonus factor for AATB)
6. A launch time window
7. A start procedure, including timing
8. Optionally, a task deadline

3.5.2 Proposal

A race task definition consists of:

1. A launch point, given as GPS coordinates
2. A number of turnpoint cylinders
3. A goal
4. An indication which of the turnpoint cylinders is the start (start of speed section)

5. If goal does not serve as end of speed section: An indication which of the turnpoint cylinders is the end of speed section.
6. A launch time window
7. A start procedure, including timing
8. Optionally, a task deadline

3.5.3 Reasoning

No more final glide decelerators

3.6 6.2 Definition of control zones

3.6.1 Status quo

Control zones are geographical areas which must be reached by the pilots in the course of a task.The three types of control zones are the turnpoint cylinder, the conical end of speed section and the goal semi-circle.

3.6.2 Proposal

-

3.6.3 Reasoning

No more final glide decelerators.

3.7 6.2.2 Definition of turnpoint cylinder

3.7.1 Status quo

A turnpoint cylinder is defined as:

1. A centre point*c*, given as GPS coordinates
2. A radius*r*, given in meters
3. An indication whether the cylinder is an “exit” or an “enter” cylinder. This defines whether the corresponding turnpoint is considered reached by a pilot when crossing the cylinder’s boundary from its inside to the outside, or from its outside to the inside.

A turnpoint cylinder is then given as the cylinder with radius *r* around the axis which cuts the x/y plain orthogonally at the cylinder’s centre point*c*. For task evaluation purposes, only the cylinder’s projection in the x/y plain is considered: a circle of radius *r* around *c*.

Note that for start cylinders (SSS), “enter” only makes sense if the following turnpoint cylinder lies within the SSS cylinder. Likewise, an “exit” only makes sense if the first turnpoint lies outside of the SSS cylinder. Currently, the start direction cannot be set within FS. Instead the program automatically scores according to this logic.

3.7.2 Proposal

A turnpoint cylinder is defined as:

1. A centre point*c*, given as GPS coordinates
2. A radius*r*, given in meters

A turnpoint cylinder is then given as the cylinder with radius *r* around the axis which cuts the x/y plain orthogonally at the cylinder’s centre point*c*. For task evaluation purposes, only the cylinder’s projection in the x/y plain is considered: a circle of radius *r* around *c*.

Note that the designation of “enter” or “exit” cylinder has been removed, to reduce a potential source of confusion and task setting errors.Whether a turnpoint is considered reached is determined either by the presence of a single tracklog point inside the turnpoint cylinder’s tolerance band, or by the presence of two consecutive tracklog points which lie on opposite sides of the turnpoint cylinder boundary (a “crossing”). The direction in which such a crossing occurs is irrelevant.

Task setters may still choose to indicate whether the start or subsequent turnpoint cylinders are “enter” or “exit”, to explain their intended task route. But pilots are not bound to those indications.

3.7.3 Reasoning

No more prescribed turnpoint direction (including start)

3.8 6.2.2 Definition of conical end of speed section

3.8.1 Status quo

A conical end of speed section is defined as:

1. A centre point *c*, given as GPS coordinates and altitude
2. An incline *i*, given as a ratio of altitude/distance to the centre point
3. A radius *r*, indicating the size of the circle resulting from the intersection between the cone and a horizontal plane at goal altitude

A conical end of speed section is given as the cone resulting from an axis of inclination *i* through the centre point *c’*, which is equal to point *c*, but has its altitude lowered by *r/i* metres compared to *c*.

The incline is chosen for each task. The default value is 1:3.5. Values suggested for use are between 1:2.5 and 1:4.

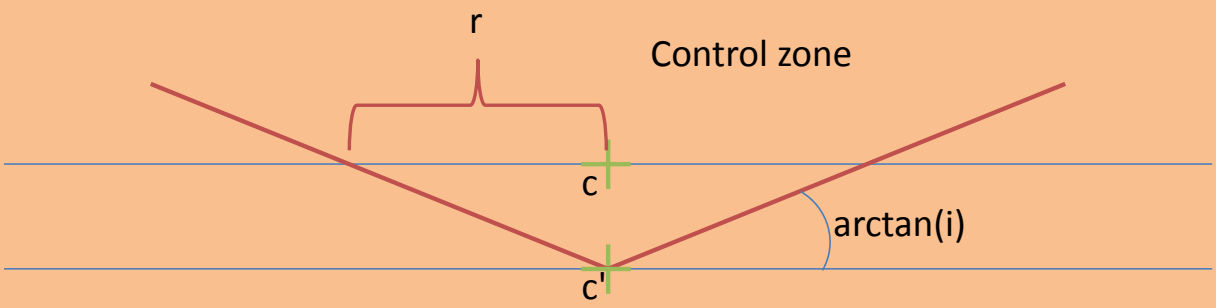

Figure 1: Conical end of speed section from the side

3.8.2 Proposal

-

3.9 Reasoning

No more final glide decelerator

3.10 6.3 Definition of goal

3.10.1 Status quo

A goal can be

1. A cylinder (enter or exit), see above, or
2. A line, defined by:
    a. A centre point*c*, given as GPS coordinates
    b. A length *l*, given in meters

If CESS is used, goal must either be a line at the cone's centre point, a cylinder at the cone's centre point with a radius smaller than radius r of the cone definition, or be located at a point that is different from the cone's centre point.

3.10.2 Proposed change

A goal can be

1. A cylinder (enter or exit), see above, or
2. A line, defined by:
    a. A centre point*c*, given as GPS coordinates
    b. A length *l*, given in meters

Note that from 2021, goal type 2 (line) will no longer be used. This because the increased use of very big turnpoint cylinders makes the turnpoint direction ambiguous. To avoid confusion, from 2021, all goals will be cylinders. We advise all task setters to start using goal cylinders exclusively as early as possible. We advise all score keepers who observe task setters setting tasks with goal lines to inform the respective task setters of this coming change.

The use of an informal goal line, to mark the goal field, will still be permitted from 2021.

3.10.3 Reasoning

No more goal lines from 2021

3.11 6.5 Distances

3.11.1 Status quo

6.5.1 Task distance

Task distance is defined as the path of shortest distance from the start point to goal that touches all turnpoint cylinders. The method to calculate this distance is buried in FS' code and will be documented at a later time.

Currently, this method mistreats the case where after ESS one or several turnpoints must be reached with centres different from ESS' centre point: The two legs to and from ESS are optimized for distance as well, which can result in a longer last leg before ESS than what pilots experience in reality. As a result, pilots can reach ESS with a shorter flown distance than what FS indicates is the distance required to reach ESS. This only affects pilots landing between ESS and goal, but does not represent a disadvantage to those pilots. The speed section distance calculation used for scoring (see below) is not affected by this.

6.5.2 Speed section distance

Speed section distance is defined as the path of shortest distance from start of speed section to end of speed section that touches all turnpoint cylinders. The method to calculate this distance is the same as for task distance.

3.11.2 Proposal

6.5.1 Route optimization algorithm

At the core of all distance calculations lies an algorithm that calculates the shortest path from a given start position through a series of turnpoint cylinders to a destination (which can be a cylinder, a goal line or a point). The algorithm used to determine this path is defined in Appendix A. The result of this algorithm is a set of points on the cylinders that represent the optimal crossing points for shortest overall distance. Distance is then calculated by iterating through those points, calculating the distance between two subsequent points (leg distance), and building the sum of those leg distances.

$routeElement = \text{point}(lat, lon) | \text{cylinder}(lat, lon, radius) | \text{goal}(lat, lon, radius)$
$routeDefinition = \{routeElement_1, \dots, routeElement_n\}$
$path = \{point_1, \dots, point_n\} \text{where} point_x = \text{point}(lat_x, lon_x)$
$\text{routeOptimization}(routeDefinition) \text{returns } path$
$\text{pathDistance}(path) = \sum_1^{n-1} \text{distance}(path.point_i, path.point_{i+1})$

6.5.2 Speed section distance

Speed section distance is defined as the sum of the individual leg distances between start of speed section and end of speed section as calculated by the route optimization algorithm.

$$speedSectionDefinition = \begin{Bmatrix} startOfSpeedSection, \\ turnpoint_1, \dots, turnpoint_m, \\ endOfSpeedSection \end{Bmatrix}$$
$speedSectionPath = \text{routeOptimization}(speedSectionDefinition)$
$speedSectionDistance = \text{pathDistance}(speedSectionPath)$

6.5.3 Task distance

Task distance is defined as speed section distance plus any distance pilots need to cover before or after the speed section:

$launch = \text{point}(lat_{launch}, lon_{launch})$

$$preSpeedSectionDefinition = \begin{Bmatrix} launch, \\ preTurnpoint_1, \dots, preTurnpoint_n, \\ speedSectionPath.point_1 \end{Bmatrix}$$
$preSpeedSectionPath = \text{routeOptimization}(preSpeedSectionDefinition)$
$preSpeedSectionDistance = \text{pathDistance}(preSpeedSectionPath)$
$$postSpeedSectionDefinition = \begin{Bmatrix} speedSectionPath.point_n, \\ postTurnpoint_1, \dots, postTurnpoint_n, \\ goal \end{Bmatrix}$$
$postSpeedSectionPath = \text{routeOptimization}(postSpeedSectionDefinition)$
$postSpeedSectionDistance = \text{pathDistance}(postSpeedSectionPath)$
$$\boldsymbol{taskDistance} = \begin{pmatrix} preSpeedSectionDistance \\ +speedSectionDistance \\ +postSpeedSectionDistance \end{pmatrix}$$

3.11.3 Reasoning

Clarify distance calculation

3.12 7 Flying a task

3.12.1 Status quo

-

3.12.2 Proposal

7.1 Re-starting

In tasks that are either set as Race to goal with multiple start gates, or as Elapsed time task, a pilot may decide to return to the start and take a later start time after already having flown a portion of the start.

A restart is possible up to the point where the first turnpoint has been reached. Returning to the start after reaching the first turnpoint has no effect on the scored start time.

3.13 Reasoning

Clarification on re-starting was requested.

3.14 8. Task evaluation

3.14.1 Status quo

8.1 Reaching a control zone

8.1.1 Reaching a turnpoint cylinder

A cylinder is considered “reached” by a pilot if that pilot’s track log shows the pilot crossing out of the cylinder in the case of an exit cylinder, or into the cylinder in case of an enter cylinder, by containing at least one track point closer to the cylinder’s centre than the cylinder radius (enter) or further away from the cylinder’s centre than the cylinder radius (exit). During task evaluation, only the x/y coordinates are considered, and a point must lie within (enter) or outside of (exit) the circle representing the turnpoint cylinder in the x/y plain. This is determined by measuring the distance between a track point and the turnpoint. This distance must be greater (exit) or smaller (enter) than the cylinder’s radius.

To compensate for the very slight distance measurement differences resulting from the use of different distance measurement algorithms, a % or minimum of 5 metre tolerance is used for this calculation. This had to be introduced so that a pilot reading the distance to the next cylinder centre from his GPS device can rely on having reached the turnpoint when the distance displayed by the instrument is smaller than the defined turnpoint cylinder radius.

For Cat 1, the tolerance is set to 0.1%.

For Cat 2, the maximum tolerance is 0.5%, to allow pilots to still use equipment that calculates distances on the FAI sphere.

For enter cylinders, this means that a tracklog point that is closer to the turnpoint than r*1.005 is considered proof of reaching the turnpoint. For exit cylinders, this means that a tracklog point that is further away from the turnpoint than r*0.995 is considered proof of reaching the turnpoint.

The time when a control zone was reached is determined by the time a so-called “crossing” occurred. A crossing is defined as a pair of consecutive track points, of which at least one lies inside the band determined by the turnpoint’s centre, its radius and the tolerance value.

$$tolerance = 0.5\%|0.1\%$$

$$minTolerance = 0|5m$$

$$turnpoint_i : innerRadius_i = \min(radius_i * (1 - tolerance), radius_i - minTolerance)$$

$$turnpoint_i : outerRadius_i = \max(radius_i * (1 + tolerance), radius_i + minTolerance)$$

$$crossing_{turnpoint_i} : \exists j : (\text{distance}(center_i, trackpoint_j) >= innerRadius_i$$

$$\wedge \text{distance}(center_i, trackpoint_{j+1}) <= outerRadius_i)$$

$$\vee (\text{distance}(center_i, trackpoint_{j+1}) >= innerRadius_i$$

$$\wedge \text{distance}(center_i, trackpoint_j) <= outerRadius_i)$$

The time of a crossing depends on whether it actually cuts across the actual cylinder, or whether both points lie within the tolerance band, but on the same side of the actual cylinder.

$$(\text{distance}(center_i, trackpoint_j) < radius_i \wedge \text{distance}(center_i, trackpoint_{j+1}) < radius_i)$$

$$\vee (\text{distance}(center_i, trackpoint_j) > radius_i \wedge \text{distance}(center_i, trackpoint_{j+1}) > radius_i)$$

$$\wedge turnpoint_i = ESS : crossing.time = trackpoint_{j+1}.time$$

$$(\text{distance}(center_i, trackpoint_j) < radius_i \wedge \text{distance}(center_i, trackpoint_{j+1}) < radius_i)$$

$$\vee (\text{distance}(center_i, trackpoint_j) > radius_i \wedge \text{distance}(center_i, trackpoint_{j+1}) > radius_i)$$

$$\wedge turnpoint_i \neq ESS : crossing.time = trackpoint_j.time$$

$$(\text{distance}(center_i, trackpoint_j) < radius_i \wedge \text{distance}(center_i, trackpoint_{j+1}) > radius_i)$$

$$\vee (\text{distance}(center_i, trackpoint_j) > radius_i \wedge \text{distance}(center_i, trackpoint_{j+1}) < radius_i) :$$

$$crossing.time = \text{interpolateTime}(trackpoint_{j+1}, trackpoint_{j+1})$$

The method used to interpolate the crossing time is buried in FS' code and will have to be documented at a later point.

Finally, given all *n* crossings for a turnpoint cylinder, sorted in ascending order by their crossing time, the time when the cylinder was reached is determined.

$$turnpoint_i = SSS : reachingTime_i = crossing_n.time$$

$$turnpoint_i \neq SSS : reachingTime_i = crossing_0.time$$

8.2 Reaching a conical end of speed section

A conical end of speed section is considered "reached" by a pilot if that pilot's track log contains at least one point where the required glide angle to the cone's centre point is equal to or smaller than the cone's incline.

$$crossing_i : \exists j : \frac{trackpoint_j.altitude - center_{CESS}.altitude}{\text{distance}(center_{CESS}, trackpoint_j)} <= incline$$

$$\wedge \frac{trackpoint_{j-1}.altitude - center_{CESS}.altitude}{\text{distance}(center_{CESS}, trackpoint_{j-1})} > incline$$

$$crossing_i.time = \text{interpolateTime}(trackpoint_{j+1}, trackpoint_{j+1})$$

Given all *n* crossings for a CESS, sorted in ascending order by their crossing time, the time when CESS was reached is determined.

$$reachingTime_{CESS} = crossing_0.time$$

If goal is at the CESS centre point, and a pilot reaches goal without previously entering the CESS, he is considered having reached CESS at the time when he crossed the goal line.

3.15 Proposal

8.1 Cylinder tolerance

To compensate for the very slight distance measurement differences resulting from the use of different distance measurement algorithms in flight recorders and evaluation programs, a tolerance, consisting of a percentage and an absolute minimum, is applied when determining whether a pilot reached a cylinder. This had to be introduced so that a pilot reading the distance to the next cylinder centre from his GPS device can rely on having reached the turnpoint when the distance displayed by the instrument is smaller than the defined turnpoint cylinder radius.

For Cat 1, the tolerance is set to 0.1%.

For Cat 2, the maximum tolerance is 0.5%, to allow pilots to still use equipment that calculates distances on the FAI sphere.

$$tolerance = 0.1\% \mid 0.5\%$$
$$minTolerance = 5m$$
$$turnpoint_i : innerRadius_i = \min(radius_i * (1 - tolerance), radius_i - minTolerance)$$
$$turnpoint_i : outerRadius_i = \max(radius_i * (1 + tolerance), radius_i + minTolerance)$$

8.2 Reaching a cylinder

To determine whether a pilot reached a specific cylinder in the task, the pilot's track log must show evidence that:

1. The pilot reached the previous cylinder in the task definition
2. It contains at least one crossing for the cylinder, defined as either of the following:
    a. Existence of a single tracklog point inside the turnpoint cylinder's tolerance band – crossing time is the time at which this tracklog point was recorded
    b. A pair of consecutive tracklog points which lie on opposite sides of the turnpoint cylinder boundary – crossing time is linearly interpolated from the two tracklog points' recording times and their respective distances from the boundary
3. It contains a valid crossing, which occurred later than the valid crossing of the previous cylinder in the task definition, but no earlier than the start time, and no later than the task deadline.

$$crossing_{turnpoint_i} : \exists j : (\text{distance}(center_i, trackpoint_j) >= innerRadius_i$$
$$\wedge \text{distance}(center_i, trackpoint_j) <= outerRadius_i)$$
$$\vee(\text{distance}(center_i, trackpoint_j) > radius_i \wedge \text{distance}(center_i, trackpoint_{j+1}) < radius_i)$$
$$\vee(\text{distance}(center_i, trackpoint_j) < radius_i \wedge \text{distance}(center_i, trackpoint_{j+1}) > radius_i)$$

The time of a crossing depends on whether it cuts across the cylinder boundary, or whether it consists of a point within the tolerance band.

$$(\text{distance}(center_i, trackpoint_j) >= innerRadius_i \wedge \text{distance}(center_i, trackpoint_j) <= outerRadius_i) : crossing.time = trackpoint_{j+1}.time$$

$$(\text{distance}(center_i, trackpoint_j) < radius_i \wedge \text{distance}(center_i, trackpoint_{j+1}) > radius_i) \vee (\text{distance}(center_i, trackpoint_j) > radius_i \wedge \text{distance}(center_i, trackpoint_{j+1}) < radius_i) : crossing.time = \text{interpolateTime}(trackpoint_j, trackpoint_{j+1})$$

The method used to interpolate the crossing time is buried in FS' code and will have to be documented at a later point.

8.2.1 Start time

In Race to Goal tasks with multiple start gates, and in Elapsed Time tasks, the last crossing of the start cylinder that is followed by a crossing of the first turnpoint cylinder (or the landing) is taken as the pilot's start, and that crossing's time is used as the pilot's start time.

3.15.1 Reasoning

Correct and improve definitions, clarify, no more final glide decelerators

3.16 8.3.2 Goal line

3.16.1 Status quo

Entering the goal control zone (semi-circle behind the goal line, see 6.3.1) from any direction without prior crossing of the goal line is equivalent to crossing the goal line.

3.16.2 Proposal

Entering the goal control zone (semi-circle behind the goal line, see 6.3.1) from any direction without prior crossing of the goal line is equivalent to crossing the goal line. To determine whether the goal control zone was reached, the same tolerance calculations apply as for full cylinders (8.1).

3.16.3 Reasoning

Clarification

3.17 10. Points allocation

3.17.1 Status quo

$$DistanceWeight = 0.9 - 1.665 * GoalRatio + 1.713 * GoalRatio^2 - 0.587 * GoalRatio^3$$

$$GoalRatio = 0 : LeadingWeight = \frac{BestDistance}{TaskDistance} * 0.1$$

$$GoalRatio > 0 : LeadingWeight = \frac{1 - DistanceWeight}{8} * 1.4 * 2$$

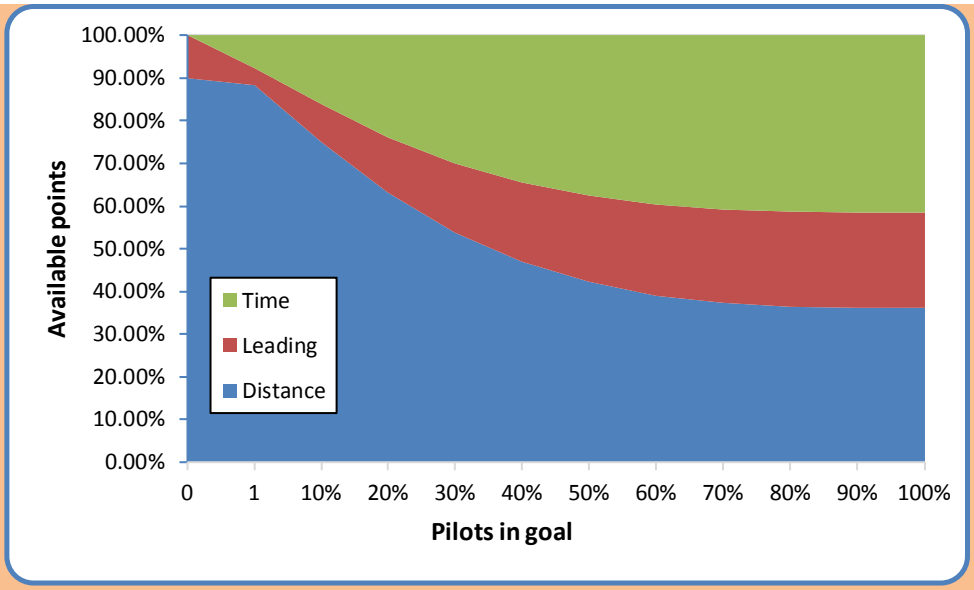

Figure 2: Points allocation for paragliding

3.17.2 Proposal

- $GoalRatio = 0: DistanceWeight = 0.838$
- $GoalRatio > 0: DistanceWeight = 0.805 - 1.374 * GoalRatio + 1.413 * GoalRatio^2 - 0.484 * GoalRatio^3$
- $LeadingWeight = 0.162$

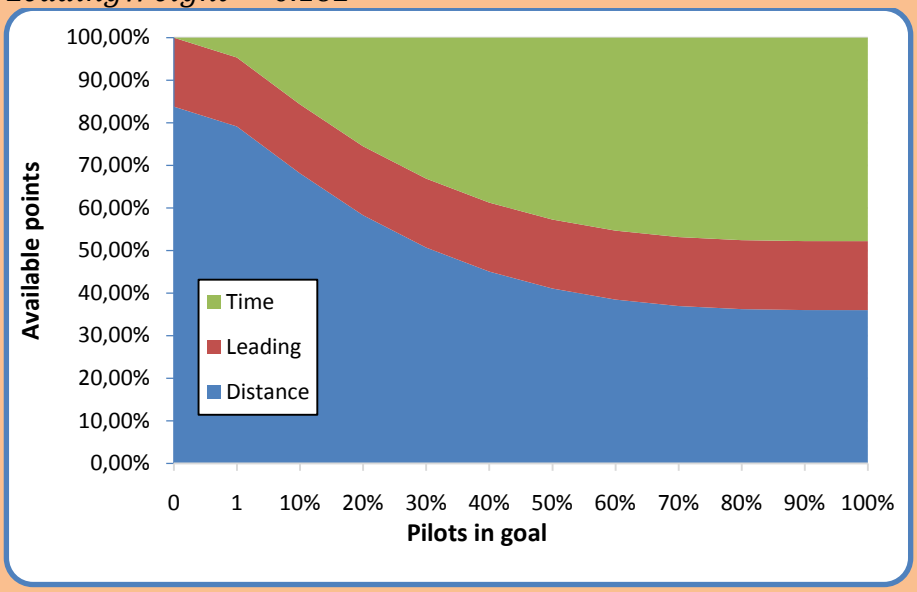

Figure 3: Points allocation for paragliding

3.17.3 Reasoning

Make only time points depend on goal ratio, approximate PWCA time points ratio.

3.18 11 Pilot score

3.18.1 Status quo

Each pilot's score is the sum of that pilot's distance, time, leading and arrival points, rounded to the nearest whole number, 0.5 being rounded up.

3.18.2 Proposal

Each pilot's score is the sum of that pilot's distance, time, leading and arrival points, rounded to one decimal place

3.18.3 Reasoning

Rounding to whole numbers only at competition score level

3.19 11.1 Distance points

3.19.1 Status quo

In the case of a stopped task, a pilot's distance may be increased by an altitude bonus (see 12.3.2). The available distance points are assigned to each pilot linearly, based on the pilot's distance flown in relation to the best distance flown in the task.

$$DistancePoints_p = \frac{Distance_p}{BestDistance} * AvailableDistancePoints$$

3.19.2 Proposal

The available distance points are assigned to each pilot linearly, based on the pilot's distance flown in relation to the best distance flown in the task.

$$DistancePoints_p = \frac{Distance_p}{BestDistance} * AvailableDistancePoints$$

In the case of a stopped task, a pilot's distance may be increased by an altitude bonus (see 12.3.2).

3.19.3 Reasoning

Stopped task remark applies to both HG and PG.

3.20 11.2 Time points

3.20.1 Status quo

$$SpeedFraction_p = \max(0, 1 - \sqrt[3]{\frac{(Time_p - BestTime)^2}{\sqrt{BestTime}}})$$

$$TimePoints_p = SpeedFraction_p * AvailableTimePoints$$

3.20.2 Proposal

$$SpeedFraction_p = \max(0, 1 - \sqrt[3]{\frac{(Time_p - BestTime)^2}{\sqrt{BestTime}}})$$

$$SpeedFraction_p = \max(0, 1 - \sqrt[6]{\frac{(Time_p - BestTime)^5}{\sqrt{BestTime}}})$$

3.20.3 Reasoning

Adopt speed fraction calculation from PWCA.

3.21 11.3.1 Leading coefficient

3.21.1 Status quo

Calculation of the leading coefficient (LC) for each pilot follows this formula:

$$LC_p = \frac{\sum_{i:tp_i \in TrackPointsInSS_p} \text{taskTime}(tp_i) * (\text{bestDistToESS}(tp_{i-1})^2 - \text{bestDistToESS}(tp_i)^2)}{1800 * LengthOfSpeedSection^2}$$

$$\forall p : p \in PilotsLandedOut \wedge \text{taskTime}(tp_{\max}) < ESSTime_{LastPilotAtESS} :$$

$$LC_p = LC_p + LastTime_{LastPilotAtESS} * \text{bestDistToESS}(tp_{\max})^2$$

$$\forall p : p \in PilotsLandedOut \wedge \text{taskTime}(tp_{\max}) \geq ESSTime_{LastPilotAtESS} :$$

$$LC_p = LC_p + \text{taskTime}(tp_{\max}) * \text{bestDistToESS}(tp_{\max})^2$$

$$\text{taskTime}(tp) = \min(TaskDeadline, \text{time}(tp))$$

$$\text{bestDistToESS}(tp_0) = LengthOfSpeedSection$$

$$\forall i : i > 0 \wedge tp_i \in TrackPointsInSS_p :$$

$$\text{bestDistToESS}(tp_i) = \min(\text{bestDistToESS}(tp_{i-1}), LenghtOfSpeedSection - \text{distanceFlown}(tp_i))$$

- In tasks where CESS is used, the CESS's centre point is considered the last point of the speed section. For LC calculations, any pilot crossing into the CESS's cone is immediately awarded the remaining distance to the cone's centre.

3.21.2 Proposal

Calculation of the leading coefficient (LC) for each pilot follows this formula:

$$LC_p = \frac{\sum_{i:tp_i \in TrackPointsInSS_p} \text{taskTime}(tp_i) * (\text{bestDistToESS}(tp_{i-1})^2 - \text{bestDistToESS}(tp_i)^2)}{1800 * LengthOfSpeed\,Section^2}$$

$$\forall p : p \in PilotsLandedOut \wedge \text{taskTime}(tp_{\max}) < ESSTime_{LastPilotAtESS} :$$

$$LC_p = LC_p + LastTime_{LastPilotAtESS} * \text{bestDistToESS}(tp_{\max})^2$$

$$\forall p : p \in PilotsLandedOut \wedge \text{taskTime}(tp_{\max}) \geq ESSTime_{LastPilotAtESS} :$$

$$LC_p = LC_p + \text{taskTime}(tp_{\max}) * \text{bestDistToESS}(tp_{\max})^2$$

$$\text{taskTime}(tp) = \min(TaskDeadline, \text{time}(tp))$$

$$\text{bestDistToESS}(tp_0) = LengthOfSpeed\,Section$$

$$\forall i : i > 0 \wedge tp_i \in TrackPointsInSS_p :$$

$$\text{bestDistToESS}(tp_i) = \min(\text{bestDistToESS}(tp_{i-1}), LenghtOfSpeed\,Section - \text{distanceFlown}(tp_i))$$

- $leadingArea_p = \sum_{i:tp_i \in trackPointsInSS_p} \text{weight}(\text{minToESS}(tp_i)) * (\text{taskTime}(tp_i) - \text{taskTime}(tp_{i-1})) * \text{minToESS}(tp_i)$
- $bestTrackPoint_p = \text{trackPointWithShortestDistanceToESS}(trackPointsInSS_p)$
- $missingArea_p = \text{weightFalling}(\text{distToEss}(bestTrackPoint_p)) * (maxTime - bestTrackPoint_p.time) * \text{distToESS}(bestTrackPoint_p)$
- $LC_p = \frac{leadingArea_p + missingArea_p}{1800 * speedSectionDistance}$

- $\text{weight}(missingDistance) = \text{weightRising}(missingDistance) * \text{weightFalling}(missingDistance)$
- $\text{weightRising}(missingDistance) = (1 - 10^{9*\frac{missingDistance}{speedSectionDistance}-9})^5$
- $\text{weightFalling}(missingDistance) = \left(1 - 10^{-3*\frac{missingDistance}{speedSectionDistance}}\right)^2$
- $\text{minToESS}(tp_0) = speedSectionDistance$
- $\forall i: i > 0, tp_i \in trackPointsInSS_p: \text{minToESS}(tp_i) = \min(\text{minToESS}(tp_{i-1}), \text{distToESS}(tp_i))$
- $\text{taskTime}(trackPoint) = \min(trackpoint.time, taskDeadline)$
- $maxTime = \min(\max(lastOutlandingTime, lastESStime), taskDeadline)$

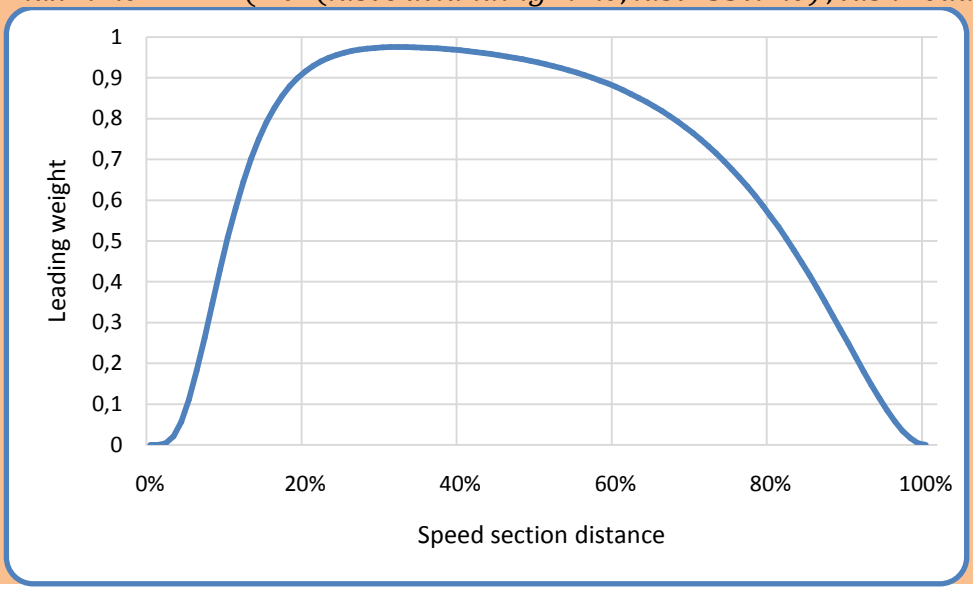

Figure 4: Leading weight in paragliding

3.21.3 Reasoning

Adopt leading weight from PWCA, no more final glide decelerators

3.22 13.3.2 Requirements to score a stopped task

3.22.1 Status quo

For a stopped task to be scored, it must fulfil certain requirements, which differ between the two disciplines:

In hang gliding, a stopped task can only be scored if either a pilot reached goal or the race had been going on for a certain minimum time. The minimum time depends on whether the competition is the Women's World Championship or not. The race start time is defined as the time when the first valid start was taken by a competition pilot.

$typeOfCompetition = Women's : minimumTime = 60\,\text{min}.$
$typeOfCompetition \neq Women's : minimumTime = 90\,\text{min}.$
$taskStopTime - timeOfFirstStart < minimumTime \wedge$
$numberOfPilotsInGoal(taskStopTime) = 0 : taskValidity = 0$

Note that this rule is currently not enforced by FS: The decision whether a stopped task is cancelled or scored must be taken by the score keeper.

In paragliding, a stopped task will be scored if the flying time was one hour or more. For Race to Goal tasks, this means that the Task Stop Time must be one hour or more after the race start time. For all other tasks, in order for them to be scored, the task stop time must be one hour or more after the last pilot started.

$minimumTime = 60\,\text{min}.$
$typeOfTask = RaceToGoal \wedge numberOfStartGates = 1:$
$taskStopTime - startTime < minimumTime : taskValidity = 0$
$TypeOfTask \neq RaceToGoal \vee numberOfStartGates > 1:$
$taskStopTime - \max(\forall p : p \in StartedPilots : startTime_p) < minimumTime : taskValidity = 0$

3.22.2 Proposal

-

3.22.3 Reasoning

Count stopped tasks regardless of duration, let validity calculation, especially stopped task validity, ensure sportive value of such tasks.

3.23 12.3.4 Time points for pilots at or after ESS

3.23.1 Status quo

Pilots who were at a position between ESS and goal at the task stop time will be scored for their complete flight, including the portion flown after the task stop time. This is to remove any discontinuity between pilots just before goal and pilots who had just reached goal at task stop time.

If a Conical ESS is used, then all pilots who have crossed into the cone before or at the task stop time will be scored for being in goal.

A fixed amount of points is subtracted from the time points of each pilot that makes goal in a stopped task. This amount is the amount of time points a pilot would receive if he had reached ESS exactly at the task stop time. This is to remove any discontinuity between pilots just before ESS and pilots who had just reached ESS at task stop time.

$typeOfTask = RaceToGoal \wedge numberOfStartGates = 1:$
$timePointsReduction = \text{timePoints}(taskStopTime - startTime)$
$TypeOfTask \neq RaceToGoal \vee numberOfStartGates > 1:$
$timePointsReduction = \text{timePoints}(taskStopTime - \max(\forall p : p \in StartedPilots : startTime_p))$

$\forall p : p \in PilotsInGoal : finalTimePoints_p = timePoints_p - timePointsReduction$

3.23.2 Proposal

Pilots who were at a position between ESS and goal at the task stop time will be scored for their complete flight, including the portion flown after the task stop time. This is to remove any discontinuity between pilots just before goal and pilots who had just reached goal at task stop time.

A fixed amount of points is subtracted from the time points of each pilot that makes goal in a stopped task. This amount is the amount of time points a pilot would receive if he had reached ESS exactly at the task stop time. This is to remove any discontinuity between pilots just before ESS and pilots who had just reached ESS at task stop time. The removed points are then redistributed as distance points.

$typeOfTask = RaceToGoal \wedge numberOfStartGates = 1:$
$timePointsReduction = \text{timePoints}(taskStopTime - startTime)$
$TypeOfTask \neq RaceToGoal \vee numberOfStartGates > 1:$
$timePointsReduction = \text{timePoints}(taskStopTime - \max(\forall p : p \in StartedPilots : startTime_p))$

$$\forall p : p \in PilotsInGoal : finalTimePoints_p = timePoints_p - timePointsReduction$$

$$AvailableDistancePointsNew = AvailableDistancePoints + timePointsReduction$$

3.23.3 Reasoning

No more final glide decelerator, redistribute removed time points

3.24 15 FTV – Fixed total validity

3.24.1 Status quo

Fixed Total Validity (FTV) is a procedure to score pilots on their best task performances, rather than all their tasks. Fixed Total Validity means the sum (total) of available points (validity) is set (fixed) to the same value for each competitor.

$$FTV\_factor = 0.2 \vee 0.25$$

$$CalculatedFTV = (1 - FTV\_factor) * \sum_{t:Task} \frac{AvailablePoints_t}{1000}$$

To calculate a pilot's FTV score, for all his flights:

1. Calculate a performance percentage for each day by dividing the pilot's day score by the day's available points
2. Arrange all flights in descending order of performance percentage
3. Total up the flights' raw day scores (not performance percentages) in order of performance percentage until the sum of validities for those scores reaches the pre-decided Fixed Total Validity value.

If the last score added takes that pilot's total validity above the Fixed Total Validity, then only a fraction of that score is used so that the pilot's total validity is equal to the Fixed Total Validity.

$$\forall t : t \in ScoredTasks : Performance_{p,t} = \frac{Score_{p,t}}{AvailablePoints_t}$$

$$SortedPerformance_p = \text{sortDescending}(\forall t : t \in ScoredTasks : Performance_{p,t})$$

$$OrderedValidities_p = \text{orderByPerformance}(SortedPerformance_p, \forall t : t \in ScoredTasks : \frac{AvailablePoints_t}{1000})$$

$$OrderedScores_p = \text{orderByPerformance}(SortedPerformance_p, \forall t : t \in ScoredTasks : Score_{p,t})$$

$$FTV\_Score_p = \sum_{u=0}^{numberOfTasks} OrderedScores_{p,u} * \max(1, \frac{\min(0, CalculatedFTV - (u > 0 : \sum_{v=0}^{\min(0,u-1)} OrderedValidities_{p,v}))}{OrderedValidities_{p,u}})$$

3.24.2 Proposal

Fixed Total Validity (FTV) is a procedure to score pilots on their best task performances, rather than all their tasks. Fixed Total Validity means the sum (total) of scored points (validity) is set (fixed) to the same value for each competitor.

$$FTV\_factor = 0.2 \vee 0.25$$

$$CalculatedFtv = (1 - FTV\_Factor) * \sum_{t:Task} \frac{WinnerScore_t}{1000}$$

To calculate a pilot's FTV score, for all his flights:

1. Calculate a performance percentage for each day by dividing the pilot's day score by the day winner's points
2. Arrange all flights in descending order of performance percentage
3. Total up the flights' raw day scores (not performance percentages) in order of performance percentage until the sum of validities for those scores reaches the pre-decided Fixed Total Validity value.

If the last score added takes that pilot's total validity above the Fixed Total Validity, then only a fraction of that score is used so that the pilot's total validity is equal to the Fixed Total Validity.

$$\forall t : t \in ScoredTasks : Performance_{p,t} = \frac{Score_{p,t}}{WinnerScore_t}$$

$$SortedPerformance_p = \text{sortDescending}(\forall t : t \in ScoredTasks : Performance_{p,t})$$

$$OrderedValidities_p = \text{orderByPerformance}(SortedPerformance_p, \forall t : t \in ScoredTasks : \frac{WinnerScore_t}{1000})$$

$$OrderedScores_p = \text{orderByPerformance}(SortedPerformance_p, \forall t : t \in ScoredTasks : Score_{p,t})$$

$$FTV\_Score_p = \sum_{u=0}^{numberOfTasks} OrderedScores_{p,u} * \max(1, \frac{\min(0, CalculatedFTV - (u > 0 : \sum_{v=0}^{\min(0,u-1)} OrderedValidities_{p,v}))}{OrderedValidities_{p,u}})$$