SubSkipper Documentation

Filip Socko

August 18, 2015

1 Introduction

The GitHub SubSkipper repository contains the core logic of the app, such that it can be verified or used for other projects. The documentation contains the principles and equations on which the logic is based, and some method documentation.

The main purpose of the repository and documentation is to record techniques and methods of early submarine attack techniques in a way which are simple to employ in computer programs (i.e. showing mathematical equations where possible), as well as acting as a reference for Submarine Simulators.

The Android App will be developed from this repository as a separate, polished product.

2 Requirements

The Requirements for SubSkipper are the following:

- Calculate Constant Bearing Solutions using the following methods:
  - Dick O'Kane method
- Show a representation of the *Is/Was* or *AngriffScheibe* calculator for easy input, along with its data
- Calculate target location with a course and speed
- Calculate *AOB* from aspect ratio
- Calculate Speed via *Fixed Wire* method
- Feature a Modular timer/stopwatch

3 Unit Conversions

3.1 Speed

1.0 knots	1.0 NM per hour
1.0 m/s	1.94384449 knots
1.0 m/s	3.6 km/h

3.2 Length

1.0 NM	1852.0 m
1.0 km	0.539956803 NM
1.0 NM	2025.37 yards
1.0 NM	6076.12 feet
1.0 m	3.2808399 feet
1.0 m	1.0936133 yards

4 Torpedo Data

Torpedo data adapted from *SH4 V1.2 Ultimate Torpedo Guide* by Mechan, found at the ubisoft forums (http://forums.ubi.com/showthread.php/475595-SH4-V1-2-Ultimate-Torpedo-Guide-Forums). Dated: 06-22-2007, Accessed: 18.08.2015

Slow and Fast speeds and ranges removed as appropriate.

4.0.1 Mark 10

Available since:	Always
Range(m):	3200
Speed(kt):	36
Warhead:	80-160 radius 3-6m
Depth keeping:	70% chance of deviating 0.8m-1.2m
Dud Chance (at AOB):	1% (0°-70°) 25% (70°-90°)
Renown cost:	0

Notes: Default torpedo for the S-Class. Slower and with a shorter range than the mk14, but extremely reliable.

4.0.2 Mark 14

Available since:	Always
Range(Slow)(m):	8200
Range(Fast)(m):	4100
Speed(Slow):	31
Speed(Fast)(kt):	46
Warhead:	100-170 radius 3-7m
Depth keeping:	70% chance of deviating 1.5m-3.3m
Dud Chance (at AOB):	1% (0°-35°); 34% (35°-70°); 99% (70°-90°)
Renown cost:	0

Notes: Default torpedo for all modern fleet boats. Faster and with a longer range than the mk10, it packs a roughly 20

4.0.3 Mark 16

Available since:	1945-01-01
Range(m):	12500
Speed(kt):	46
Warhead:	180-250 radius 3.5-8m
Depth keeping:	70% chance of deviating 1.5m-3.3m
Dud Chance (at AOB):	4% (0°-35°); 45% (35°-70°); 100% (70°-90°)
Renown cost:	400

Notes: Fast torpedo with an exceptionally long range, but also terribly unreliable

4.0.4 Mark 18

Available since:	1943-07-12
Range(m):	3650
Speed(kt):	29
Warhead:	120-180 radius 3-7
Depth keeping fault chance:	55% chance of deviating 1.2m-2.8m
Dud Chance (at AOB):	1% (0°-35°); 34% (35°-70°); 99% (70°-90°)
Renown cost:	500 (200 from 1944-01-16; 0 from 1944-09-01)

Notes: Slower, 10% more powerful, with a shorter range and much more reliable than the Mk. 14.

4.0.5 Mark 23

Available since:	1943-01-01
Range(m):	4100
Speed(kt):	46
Warhead:	120-180 radius 3-7m
Depth keeping fault chance:	70% chance of deviating 1.5m-3.3m
Dud Chance (at AOB):	1% (0°- 35°); 34% (35°- 70°); 99% (70°- 90°)
Renown cost:	100 (0 from 16-01-1944)

Notes: Same range, speed and reliability than the Mk. 14 but roughly 10%

more powerful. Definitely replaces the mk10 as the "standard" torpedo from 16-01-1944.

4.0.6 Mark 27 "Cutie"

Available From:	1944-01-01
Range(m):	4570
Speed(kt):	12
Warhead:	50-100 radius 1.5-5
Depth keeping fault chance:	NA
Dud Chance (at AOB):	1% (0°-25°)
Renown cost:	500

Notes: Slow acoustic homing torpedo with a small warhead primarily used for defense against destroyers.

5 O'Kane Torpedo Solution

The Dick O'Kane method was devised by members of the Subsim.com forums. It is a constant bearing method which relies on calculating a lead angle – an angle on which torpedoes, if launched will intercept the course of the target– to which the periscope is pointed. As parts of the target ship cross the bearing, torpedoes are fired along it. The O'Kane method relies on being ahead of the target, and the final AOB – at which the torpedo strikes the target– to be 90°.

Calculates *lead angle* based on target and torpedo speed.
The solution requires submarine to be ahead of target.
Captain inserts target speed into TDC, puts the scope on the lead bearing, fires as the target crosses the bearing.

The Equation for lead angle is as follows:

$$LeadAngle = 90 - \arctan\left(\frac{TorpedoSpeed}{TargetSpeed}\right)$$

5.1 Computational Solution

The method *oKSolution()* in the class *OKane.class* in the package *coreLogic* is used to calculate the O'Kane Lead Bearing when using the O'Kane method. The periscope is pointed to the lead bearing calculated by *oKane()*.

The method takes the following arguments:

- int AOB - For determining whether AOB is Port or Starboard
- double targS - Target Speed
- Torpedo fireS - Torpedo Speed

5.2 Errors

Attempting to compute O'Kane Lead angle given the following situations will return the flag -1.

5.2.1 Submarine is not ahead of target:

The following code checks if AOB is ahead of the target, either port or starboard. Furthermore, if the submarine is at an AOB of 0, or 180, the O'Kane lead angle cannot be calculated.

```
if(AOB <= 90 && AOB<360){
            stbd = true;
        }
        else if(AOB>=270 && AOB<360 ){
            stbd = false;
        }
        else if(AOB == 0){
            invalidSol = true;
```

5.2.2 Torpedo Speed or Target Speed are less than one:

torpS and targS are verified to be ¿ 1 in the following code:

```
 if(torpS < 1 || targS <1){
            return -1;}
```

5.2.3 Lead is greater than 90

If lead is more than 90°, it means we would be aiming the torpedo backwards. This means the solution is invalid, as either the target is too fast, or the torpedo too slow. This error is handled in the following code:

```
if(lead > 90){
lead = -3;} //Okane relies on being ahead of the target,
                  //we would be aiming backwards.
```

5.2.4 Target Speed is 0

If targS is 0, no lead is required and the method returns 0. This error is handled in the following code:

```
//If speed is 0, no lead required
            if(targS == 0){
               return 0;
            }
```

5.3 OKSolution Code

```java
public double OKSolution(int AOB, double targS, double torpFireS)
{
    double solBearing = -1;

    boolean stbd = true;
    //check if the position is correct. Sub needs to be in front
        of target, on either
    //Stbd or port side. this means AOB is either 0-90 for stbd
        or 270-360 for port
    //if anything else happens, solution is invalid, and we
        return a flag.
    boolean invalidSol = false;

 if(AOB == 0){
        invalidSol = true; //for now let's assume the user is an
            idiot if AOB = 0
 }
 else if(AOB <= 90 && AOB<360){
        stbd = true;
    }
    else if(AOB>=270 && AOB<360 ){
        stbd = false;
    }

    else{invalidSol = true;}

    if(invalidSol){
        return solBearing;
    }

    //Check if OKaneLead returns an error
    double lead = okaneLead(torpFireS, targS);
    if(lead == -3){
       return -1;
    }

    //if stbd, subtract from 360.
    else if(stbd){
        solBearing = 360-okaneLead(torpFireS, targS);
    }
    //If port, add to 0 for lead bearing.
    else{
        solBearing = 0 + okaneLead(torpFireS, targS);
    }
    return solBearing; //tSpeed into TDC, set your scope to this
        bearing, fire
```

```java
}

//Input: torpedo speed (kn), target speed (kn)
private double okaneLead(double torpS, double targS){
    if((torpS < 0) || (targS <0)){
        return -3;}
    //If speed is 0, no lead required
    if(targS == 0){
       return 0;
    }
    double lead = 0;
    //90 - inverseTan(torpS/targS)
    lead = 90-Math.toDegrees(Math.atan(torpS/targS));
    if(lead > 90){
        lead = -3;} //Okane relies on being ahead of the target,
    //we would be aiming backwards.

    return lead;
}
```

6 Calculating Distance To Target

No methods for calculating Distance To Target will be provided as solutions such as a periscope stadimeter and *sonar* are readily available.

7 Calculating AOB Based on Aspect Ratio

AOB can be determined given the following data:

- Range To Target
- Observed Mast Height
- Observed Ship Length
- Reference Aspect Ratio (i.e. $\frac{ReferenceLength}{ReferenceMastHeight}$)

7.0.1 Determine an observed *aspect ratio.*

$$AR_{observed} = \frac{ObservedLength}{ObservedMastHeight}$$

As the required figure is a ratio, it does not matter in what units the figures are given. For example, this could be the number of degrees Length and Mast Heigh subtend, the number of periscope graduations subtended or angular length in metres. It only matters that both units are the same.

7.0.2 Determine the Reference Apsect Ratio

Identify the target and find the Length and Mast Height as given in the recognition manual (if possible) or calculate these figures using the SubSkipper ship parser. Proceed as for the observed aspect ratio to get the Reference Aspect Ratio (*ARreference*).

7.0.3 AOB calculation

$$AOB = arcSin \frac{AR_{observed}}{AR_{reference}}$$

- Note: This method is less accurate as AOB approaches 0.
- Note: This method does not compute whether the AOB is on the port or starboard side.
- Note: "The AOB can only go up to 90, and gives no indication of starboard or port side showing. You have to determine that visually. If the target is moving away from you, you have to subtract the given angle from 180." -UJagd Tools