# Sector Congestion Analytical Modeling Program (SCAMP)

# and the Standard Index of Sector Congestion (SISCO)

William Knecht, Lauren Murphy, and Kip Smith

Human Factors and Experimental Economics Laboratory

Kansas State University

December 18, 2000

Prepared for:

The Office of the Chief Scientist and Technical Officer for Human Factors of the Federal Aviation Administration, AAR-100

## Abstract

This document presents a novel approach for modeling sector congestion called the Sector Congestion Analytical Modeling Program (SCAMP) and the congestion metric it produces – the Standard Index of Sector COngestion (SISC0). SCAMP is an airspace modeling program that calculates a component congestion metric for each pair of aircraft in a given volume of airspace. This document describes the theoretical basis for SCAMP and an experiment that validates the utility of the SISCO metric. The appendices present mathematical derivations of the SISCO metric, the SCAMP source code.

## Introduction

Human factors analyses of air traffic control puts a great deal of emphasis on controller workload (Arad, 1964; Couluris & Schmidt, 1973; Empson, 1987; Endsley & Rodgers, 1997; Kinney, et al., 1977; Langan-Fox & Empson, 1985; Stager & Hameluck, 1990; Stein, 1985). Two major factors affect controller workload: (1) the physical state, or geometric arrangement, of the airspace and, (2) the actual physical work the controller has to do to manage that airspace. We use the term *traffic congestion* (or more simply *congestion*) to refer to the purely geometric aspects of traffic flowing through some mathematically specified volume of airspace (e.g., a sector). Factors contributing to congestion include, but are not limited to, the sector shape and size, the positions, headings, speeds, and altitudes of all aircraft, and sources of uncertainty such as weather (Hopkin, 1995; Smolensky & Stein, 1998).

Traffic management units (TMU) currently count the number of aircraft in a sector and use this number (which we can call $n_a$) as their index of traffic congestion. While useful to a degree, $n_a$ does not take into account sector geometry, the vectors of individual aircraft, terrain, and weather. It simply relies on the rationale that the raw count of aircraft should correlate with workload. That correlation is not perfect (e.g., Endsley & Rodgers, 1997). Having twice the number of aircraft does not always mean twice the work for the controller. Workload depends not only on number, but also on where the aircraft are and on

the direction they are traveling relative to (a) each other, (b) the sector boundaries, and (c) other factors such as weather within the sector (Buckley, et al., 1983; Couluris & Schmidt, 1973; Empson, 1987; Mogford, et al., 1995; Rodgers, et al., 1993; Schroeder, 1982; Stagar, et al., 1989).

From a historical point of view, count was a logical way to start describing congestion, particularly given the limitations of computation power during the early years of air traffic control. Given the latest developments in information technology, the cost of computing has dropped drastically, opening up the possibility for an improved congestion metric—one which captures more features of the airspace than the raw count of aircraft. The Sector Congestion Analytical Modeling Program (SCAMP) and the congestion metric it produces--the Standard Index of Sector Congestion (SISCO)—are being developed to provide a better indicator of traffic congestion and, indirectly, of controller workload.

### What are SCAMP and SISCO?

SCAMP is a modeling program written in the scientific visualization language *Mathematica* (Wolfram Research, 1999). *Mathematica's* forte is solving and graphing mathematical equations, enabling us to create a model and easily visualize exactly how its equations behave.

SCAMP calculates a component congestion metric for each pair of aircraft. SCAMP has been theorized, programmed, and validated and is described in this report. The next step in the development process will

be to modify SCAMP to create an overall sector metric of congestion, SISCO. SISCO is currently under conceptual development. SISCO will take the pairwise congestion metrics calculated by SCAMP and combine them into a single, overall metric of sector-wide traffic congestion.

## Modeling congestion

### What is modeling?

Modeling is the technique of creating a simplified version of some complicated system. Not every feature of a complicated system is all that important to how the system behaves. So what we try to do is pick the most important features of the situation and include them in the model, while excluding things which are less important to the overall behavior of the system. The goal is to create the simplest possible system which is at the same time *accurate, reliable*, and *useful*.

In the jargon of statistics, the purpose of modeling is to account for the maximum amount of variance in some dependent variable while using the minimum number of independent variables to do it. For example, the congestion metric $n_a$ mathematically derives from a model which assumes that aircraft count is the single and only important factor in air traffic congestion. This independent variable ($n_a$--aircraft count) is supposed to account for a large proportion of the variance in the outcome variables (e.g., controller performance and workload). The trouble is that, while $n_a$ does account for *some* of the variance in workload, it is based on too simple a model (Buckley, et al., 1983; Fowler, 1980; Mogford, et al.,

1993; Mogford, et al., 1991; Schmidt, 1976; Siddiqee, 1973). That model was about as complex as older computers were capable of handling, but that situation has changed. Modern computers can easily support far more complicated (and hopefully better) models.

We want to build a more accurate, reliable, useful model of congestion. Yet we still do have to simplify things to some extent because we cannot have a model that takes *everything* into account. So the primary theoretical challenge in coming up with a congestion metric lies in coming up with a model of the airspace that is complex enough to be realistic, yet simple enough to run on a modern computer. This is a difficult problem. And, as with all difficult problems, the solution begins with decomposing the big problem into small, solvable units (Simon, 1969/1981).

## What is congestion?

The very first thing that has to be done is to define clearly and precisely what we mean by traffic “congestion”.

*Congestion is the degree to which maneuver is constrained for each aircraft in the airspace*.

This is a very practical definition, since it is based on our common-sense notion of how it feels to actually *be prevented from doing something we need to do*. It is also a very powerful definition, because it sets up the problem of congestion to be the simpler matter of figuring

out what it is in the airspace that would keep pilots from flying in whatever manner they please.

Aircraft count alone does not fully describe congestion. Consider the situation in Figure 1. Both situations 1a and 1b have three aircraft. But which situation is truly more congested?

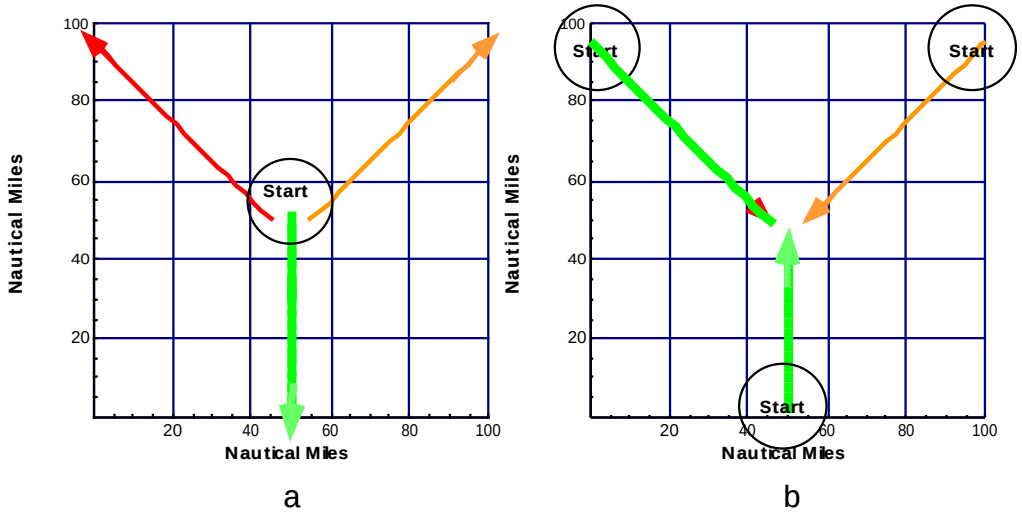



**Fig. 1. Two situations where aircraft count is identical but congestion is not. In 1a, all start in the middle and travel away from each other. There is no congestion. In 1b, all start far apart and converge to the same spot. There is extreme congestion, meaning that aircraft will eventually have to be diverted by a controller from their original, preferred direction.**

This mundane demonstration illustrates the point that congestion *is* maneuver constraint. Now all we have to do is to (1) define precisely what "maneuver constraint" is, (2) find some way to quantify it for each aircraft, (3) find some way to take the sum total of individual maneuver constraints, and (4) use them to create a global measure of total airspace maneuver constraint. We can then assume that the more overall

maneuver constraint there is in the airspace, the harder a sector will be to manage, and the greater the controller workload will be.

## What is maneuver constraint?

Our operational definition of maneuver constraint is *the degree to which changes in (1) heading, (2) speed, and (3) altitude are constrained for an aircraft at or during some defined period in time*.

This definition is based on a concept we have chosen to call the *maneuver space*. As shown in Figure 2, the maneuver space is a three-dimensional representation of *the maneuvers an aircraft can physically make, given a specified period of time*.

The maneuver space has one axis for heading, one for speed, and one for altitude. So what Figure 1 shows us is literally how much the aircraft *could* turn, how far it *could* climb and descend, and how much it *could* speed up or slow down, if it had four minutes to do it in. Given four minutes this is "the space of all possible maneuvers"—the maneuver space.

The maneuver space shows what the aircraft *can* do, given some length of time. Next we define maneuver *constraint* as what the aircraft *cannot* do in the same period of time. To estimate this we use conflict probe technology to mathematically calculate to what degree, if any, this aircraft would come unacceptably close to other objects in its maneuver space. These objects could be other aircraft, terrain, or weather; it does

not matter. Anything that would result in such a conflict would naturally constrain maneuver. And any geometric areas, or portions, of the maneuver space—combinations of heading, speed, and altitude--that would result in conflicts, we can therefore call *conflict regions*.

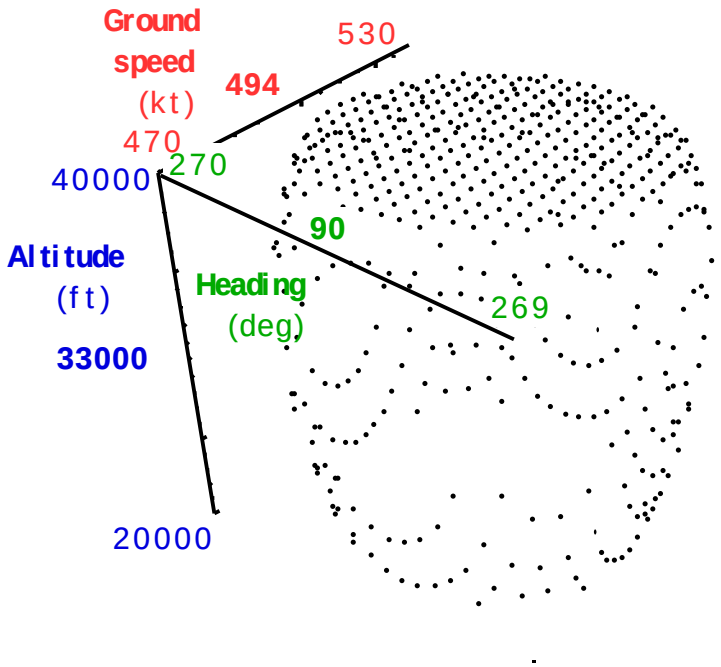



**Fig. 2. The maneuver space for an aircraft travelling 494 knots at 33000 ft (.85 mach) on a heading of 90 degrees. The lookahead time is four minutes. Note that the aircraft cannot physically climb higher than 40000 ft., but has essentially little or no restriction in descent or heading. Heading starts at the left and ends at the right in the same number because the heading dimension "wraps around" through zero (i.e. 0° = 360°).**

Consider Figure 3, which shows an airspace with two aircraft moving in it. The "ownship" is vernacular for "pilot's own ship". This scenario will result in two aircraft coming too close together. Given their vectors, a conflict probe would predict that they will simultaneously come within 1000' vertically and five miles laterally, violating each other's federally

mandated "protected zone" within a few minutes. Translating this scenario into our maneuver space/conflict region representation gives us Figure 4.

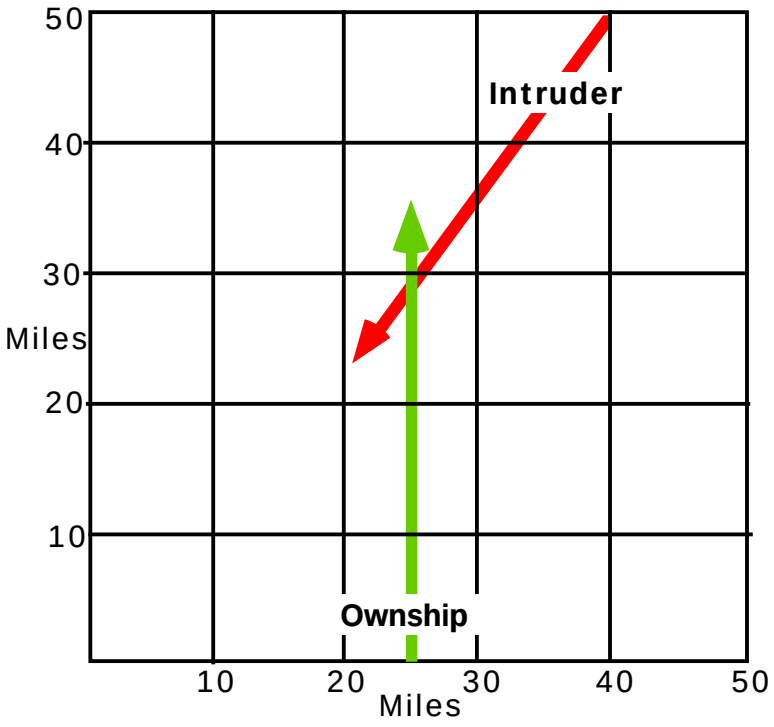



**Fig 3. Map view of a prototype conflict. Both aircraft start at a range of 50 miles and proceed at 494 knots ground speed at 33000 ft. (.85 mach), flying level, resulting in an X-crossing and violation of their protected zones.**

The conflict probe is existing technology (e.g., Batiste et al. 1997). It does, in fact, involve modeling on its own right, and can be made relatively simple or extremely complicated, depending on how accurate we want its predictions to be. Figure 4 was generated using a relatively simple probe that was derived mathematically and coded into *Mathematica*. The equations can be found in Appendix One.

Even this "simple" probe does require a fair amount of numerical computation just to draw the conflict region in Figure 3. But the point is that the logic behind the maneuver space and conflict region makes sense and is feasible in real time using technology available off-the-shelf now. What we purchase with this currency of computation is conceptual power—the ability to represent staggeringly complicated airspaces with a single, easily understandable picture.

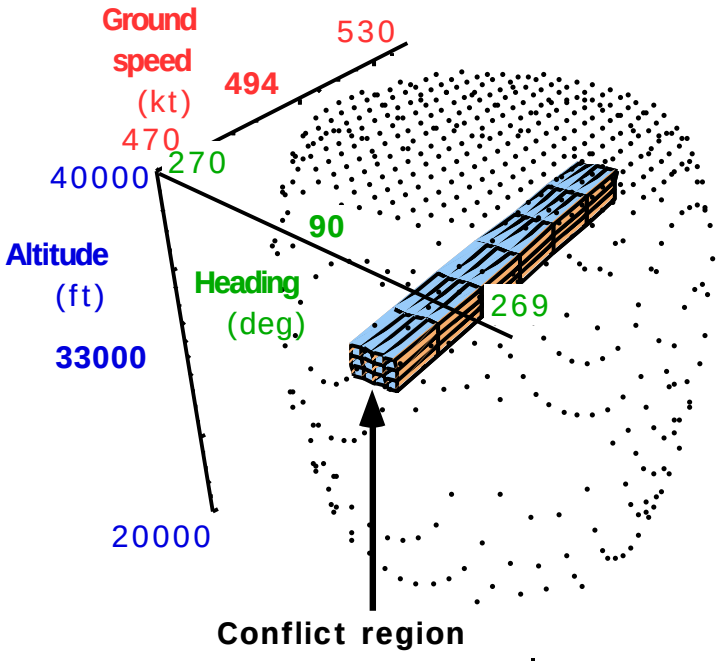



**Fig 4. Maneuver space and conflict region for the scenario of Figure 3. The maneuver space is where the ownship *could* go, physically, given four minutes. The conflict region is where it *cannot* go, if its pilot wants to avoid a conflict with the intruder aircraft.**

The maneuver space/conflict region representation is the zenith of conceptual simplicity and power. As it turns out, no matter how many elements in an airspace conspire to constrain an aircraft's maneuver,

what happens is that the maneuver space simply becomes more crowded with conflict regions. And what we can do is conceptualize some type of ratio between the occupied space and the unoccupied space, or some variation on this theme, as a literal measure of how constrained the airspace is for that particular aircraft.

## Validating SCAMP's model and its metric

SCAMP calculates a congestion metric for a single flying through a known airspace. It allows the user to specify the size of a rectangular prototype sector and to specify the initial positions, speeds, and headings for a pilot's own ship (ownship) and up to three other intruder aircraft, on the assumption that no is likely to ever experience more than three simultaneous conflicts in the en-route environment. These intruders can be vectored to conflict with the ownship simultaneously, spread out over time, or not at all to create virtually any kind of conflict.

The congestion metric itself is based on the sensible assumption that *congestion not requiring maneuver is less important than congestion that does*. In other words, if there is traffic in the airspace that I must maneuver to avoid, from my point of view, that is more "congestive" than if the same number of were present but in a geometric configuration requiring no maneuver on my part.

Another way to put this is that congestion is *factor-weighted*. In technical terms we can say congestion is weighted by discretizing the maneuver space into cells and applying a three-dimensional Gaussian

(normal curve) function—i.e. by applying one Gaussian to each axis of the maneuver space centered on the ownship's current heading, speed, and altitude--and integrating the area under each curve from the beginning of every conflict cell to its end. Since the full convolution and integration of an n-dimensional Gaussian is equal to the same value produced by the integration of a single Gaussian—1.0—the value of the pairwise congestion component is thereby also constrained to lie between 0.0 and 1.0.

It is not necessary to be a mathematician to fully understand what this means. The central concept is simply that the metric is *center-weighted*. It counts congestion around an aircraft's current heading, speed, and altitude more than it does congestion at more extreme combinations of these things. That is how it captures the "logic of conflict" that any air traffic controller will tell you makes sense in the real world: "If I don't have to move it, things are less congested than if I do."

This is how SCAMP will eventually model and calculates the global airspace congestion metric, SISCO. Now we should ask ourselves if this is valid. Does it do a better job of explaining actual data gathered from human beings than $n_a$, the simple aircraft count? SCAMP is obviously more "expensive" to compute. What, exactly, do we getting for the added expense?

One of the most straightforward, standard ways to rate an index like this is to study its criterion-referenced validity. First we find some

quantifiable measure that we can rationally accept as a valid criterion of traffic management performance. Then, given this criterion, we can calculate two correlations--one between it and SCAMP and another between it and $n_a$. As $n_a$ goes *up*, traffic management performance should go *down*, producing a negative Pearson product-moment correlation *(r)*. As SCAMP goes up, traffic management performance should also go down, also producing a negative correlation. Then the metric with the bigger correlation is the one we can assume is better related to our criterion of traffic management performance. In statistical terms one measure better explains the variance in the data set. Statistically, we call the raw correlation *r* and the percentage of data variance it explains $r^2$. The bigger $r^2$, the better the congestion metric.

The SCAMP sub-metric was evaluated using the criterion *Rmin*, which represents "minimum range achieved between two aircraft in simulated free flight" (Knecht & Hancock, 1999). Regressing *Rmin* onto $n_a$ (Figure 5a) and SCAMP (5b) show that the maneuver space-based component SCAMP metric predicts 50% more variance in *Rmin* than does $n_a$.

To obtain the data shown in Figure 5, approximately 50 commercial airline pilots flew a total of 344 flight simulator runs involving a variety of free flight X-crossings and merge-to-path situations. In these simulations, pilots had an all-white-symbology cockpit display of traffic information and were responsible for maintaining their separation with other aircraft. The dependent measure was "minimum range", *Rmin*, the minimum

geometric distance measured between the pilot's aircraft and any intruder during the course of the scenario.

This study compares the two different metrics and how well they explain operator performance in a complex ATC-relevant task. As shown in the figure on the left, *n* explains a fair amount (16%) of operator variability. However, the SCAMP sub-metric explains half again as much (24%).

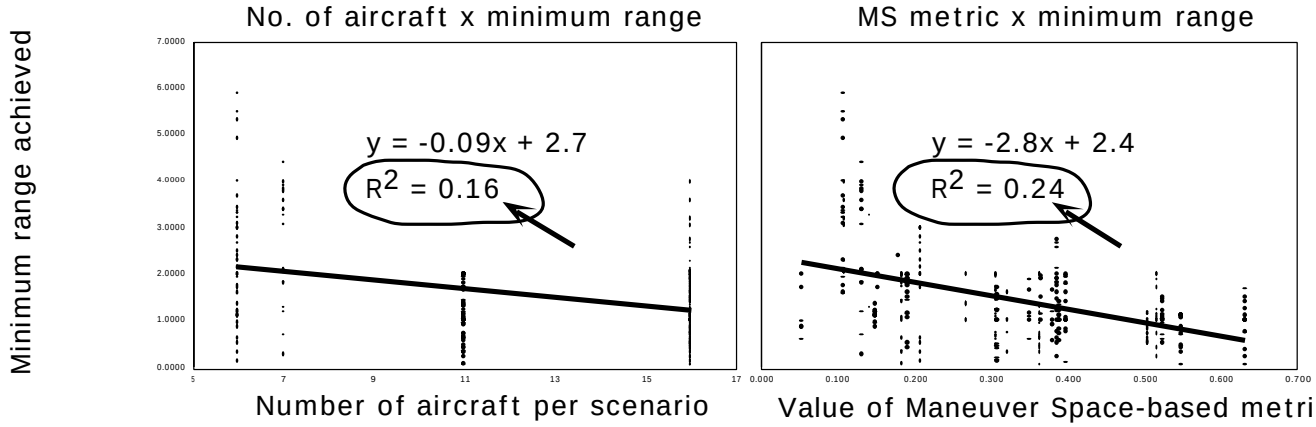



**Fig 5. Comparison of criterion-related validity for number of ($n_a$) vs. the maneuver space-based (SCAMP) sub-metric. Regression plots demonstrate how SCAMP's maneuver space-based metric predicted .24/.16 = 1.5 = 50% more variance in minimum range-per-scenario than did count.**

This test reveals that the geometric representations of the maneuver space and conflict regions are a much more powerful way to conceptualize congestion than simple aircraft count.

## Discussion

### Where do we go from here?

The first part of this effort is finished. A representational theory of congestion has been advanced, operationalized, and validated.

What remains to be done is to combine SCAMP's sub-metrics into a global metric, SISCO. This involves a second round of modeling having to do primarily with how to factor-weight and combine the value of each of the sub-metrics. This weighting could take several forms. We are considering three primary variants: (1) linearly weighting each sub-metric, (2) non-linearly weighting them (i.e. putting a disproportionately high value on the biggest sub-values), or (3) only counting values that exceed some arbitrary threshold.

All these models could be validity tested, just like the sub-metric. This would best be done by having air traffic controllers manage a number of representative traffic scenarios, much the same as our pilots managed the free flight scenarios. Then we could correlate the results (e.g. minimum range-per-scenario) with SISCO. One of the models would produce the highest correlation between SISCO and the evaluation criterion, and we could simply select that model as the one which does the best job of predicting actual controller performance, given those traffic situations.

**How will this be useful to air traffic control?**

The purpose of the TMU is to manage sector controller workload (Smith, 1999; Smith and Murphy, 2000), that is, to help modulate, or smooth out, traffic flow to individual sectors. To accomplish this, TMUs rely on count as the predictor of situational difficulty.

So far, our research indicates that we should be able to deliver a product that is conservatively *at least* half again as good at predicting congestion as the count metric used now. And if we look at what it takes to make a "just-noticeable difference" in technology these days, we see that only a 5-8%% improvement seems to typically warrant launching a new product line, at least in consumer technology. The promise that SISCO offers appears to exceed any reasonable threshold we can imagine as far as incremental improvement of air traffic management technology goes.

## References

Arad, B.A. (1964a). The control load and sector design. *Journal of Air Traffic Control*, pp. 13-31.

Batiste, V., Johnson, W., Delzell, S., Holland, S., Belcher, S., & Jordan, K. (1997). Development and demonstration of a prototype free flight cockpit display of traffic information. *Proceedings of the 1997 SAE/AIAA World Aviation Congress*.

Buckley, E.P., DeBaryshe, B.D., Hichner, N., & Kohn, P. (1983). *Methods and measurements in real-time air traffic control system simulation* (Report No. DOT/FAA/CT-83/26). Atlantic City, NJ: Federal Aviation Administration.

Couluris, G.J., & Schmidt, D.K. (1973). Air traffic control jurisdictions of responsibility and airspace structure. In *Conference on Decision and Control, 4th Annual Symposium on Adaptive Processes* (pp. 655-8). New York: Institute of Electrical and Electronics Engineers.

Empson, J. (1987). Error auditing in air traffic control. *Information systems: Failure analysis; Proceedings of the NATO Advanced Workshop*. (pp. 191-198). New York: Springer Verlag.

Endsley, M.R., & Rodgers, M.D. (1997). *Distribution of attention, situation awareness, and workload in passive air traffic control task: Implications for operational errors and automation* (Report No. DOT/FAA/AM-97/13). Washington, D.C.: Federal Aviation Administration.

Fowler, F.D. (1980). Air traffic control problems: A pilot's view. *Human Factors*, *22(9)*, 645-53.

Hopkin, V. D. (1995). *Human Factors in Air Traffic Control*. London: Taylor and Francis.

Kinney, G.C., Spahn, J., & Amato, R.A. (1977). The human element in air traffic control: Observations and analyses of the performance of

controllers and supervisors in providing ATC separation services. (Report No. MTR-7655). McLean, VA: METREK Division of the MITRE Corporation.

Knecht, W.R., & Hancock, P.A., (1999). *Separation maintenance in high-stress free flight using a time-to-contact-based cockpit display of traffic information.* Proceedings of the Human Factors and Ergonomics Society Forty-third Annual Meeting (pp. 26-29). Santa Monica: Human Factors and Ergonomics Society.

Langan-Fox, C.P., & Empson, A.C. (1985). "Actions not as planned" in military air traffic control. *Ergonomics, 28(11)*, 1509-21.

*Mathematica* V4.0, (1999). Champaign, IL: Wolfram Research.

Mogford, R.H., & Tansley, B.W. (1991). *The importance of the air traffic controller's mental model*. Paper presented at *the Human Factors Society of Canada* Annual Meeting, Canada.

Mogford, R.H., Guttman, J.A., Morrow, S.L., & Kopardekar, P. (1995). *The complexity construct in air traffic control: A review and synthesis of the literature. (*Report No. DOT/FAA/CT-TN95/22). Atlantic City, NJ: Federal Aviation Administration.

Mogford, R.H., Murphy, E.D., Yastrop, G., Guttman, J.A., & Roske-Hofstrand, R.J. (1993). *The application of research techniques for documenting cognitive processes in air traffic control*. (Report No. DOT/FAA/CT-TN93/39). Atlantic City, NJ: Federal Aviation Administration.

Rodgers, M.D., & Nye, L.G. (1993). Factors associated with the severity of operational errors at air route traffic control centers. In M.D. Rodgers (Ed.) *An examination of the operational error database for air route traffic control centers* (Report No. DOT/FAA/AM-TN93/22). NTIS No. ADA275986. Washington, D.C.: Federal Aviation Administration.

Rodgers, M.D., Mogford, R.H., & Mogford, L.S. (1998). *The relationship of sector characteristics to operational errors* (Report No. DOT/FAA/AM-98/14). Washington, D.C.: Federal Aviation Administration.

Schmidt, D.K. (1976). On modeling ATC workload and sector capacity. *Journal of Aircraft*, *13*(7), 531-7.

Schroeder, D. J. (1982). The loss of prescribed separation between aircraft: How does it occur? *Proceedings (P-114), Behavioral Objectives in Aviation Automated Systems Symposium* (pp. 257-69). Washington, D.C. Society of Automotive Engineers.

Siddiqee, W. (1973). A mathematical model for predicting the number of potential conflict situations at intersecting air routes. *Transportation Science*, *7(2)*, 158-67.

Simon, H. A. (1969/1981). Sciences of the Artificial. Cambridge, MA: MIT Press.

Smith, K. (1999). Information Requirements for Traffic Flow Management. Kansas State University Report 98-G-013.

Smith, K., & Murphy, L. (2000). TMU Structure, Positions, and Uses of the TSD. Kansas State University Report 99-G-020-2.

Smolensky, M. W., & Stein, E. S. (1998). *Human Factors in Air Traffic Control*. San Diego, CA: Academic Press.

Stager, P., & Hameluck, D. (1990). Ergonomics in air traffic control. *Ergonomics, 33,* 439-9.

Stager, P., Hameluck, D., & Jublis, R. (1989). Underlying factors in air traffic control incidents. *Proceedings of the Human Factors Society 33rd Annual Meeting* (pp. 43-6), Vol. 2. Santa Monica, CA: Human Factors Society.

Stein, E. (1985). *Air traffic controller workload: An examination of workload probe* (Report No. DOT/FAA/CT-TN84/24). Atlantic City, NJ: Federal Aviation Administration.

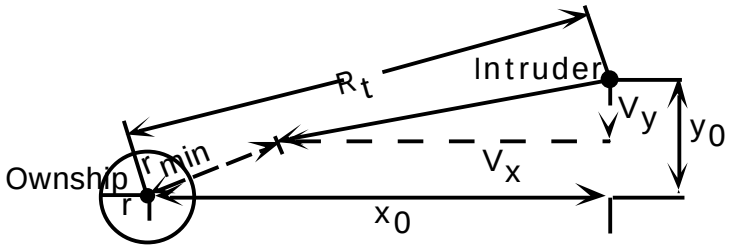
KSU HFEEL Report 99-G-020-4

## Appendix 1. Table of Contents



### 1. Derivation of the two-dimensional case of $t_{min}$

The two-dimensional case of $t_{min}$, the time-to-minimum range, or time-to-point of closest approach between the ownship and an intruder, where the coordinate system is centered on the ownship and moves along with it:




Here, $x_0$ is the relative x-distance between the ownship and the intruder at time $t_0$, $y_0$ is the relative y-distance at time $t_0$, $V_x$ is the vector component of relative velocity in the x-axis, $V_y$ is the vector component of relative velocity in the y-axis, $t$ is some given time, and $r$ is the lateral *(xy)* radius of the protected zone. Note that we are first describing the special case where $r = 0$.

**Appendix 1**

Derivation of Standard Index of Sector Congestion (SISCO)



KSU HFEEL Report 99-G-020-4

The separation distance (range) $R_t$ at any given time t is:

$$\mathrm{R_t} = \sqrt{\left(x_0 + V_x t\right)^2 + \left(y_0 + V_y t\right)^2} \tag{1.1}$$

$$= \sqrt{x_0{}^2 + 2x_0 V_x t + V_x^2 t^2 + y_0{}^2 + 2y_0 V_y t + V_y^2 t^2} \tag{1.2}$$

$$= \sqrt{\left(x_0{}^2 + y_0{}^2\right) + 2t\left(x_0 V_x + y_0 V_y\right) + t^2\left(V_x^2 + V_y^2\right)} \tag{1.3}$$

$$= \sqrt{c + bt + at^2} \tag{1.4}$$

where $c = x_0{}^2 + y_0{}^2$, (1.5)

$b = 2\left(x_0 V_x + y_0 V_y\right)$, and (1.6)

$a = V_x^2 + V_y^2$ (1.7)

To find the time $t$ where $R_t$ is minimum, we take the derivative, set it to zero, and solve for $t$:

$$0 = \sqrt{c + bt + at^2}' , \tag{1.8}$$

First noting the "trick" that the minimum of the square root of any minimizable function $f$ will be the same as the minimum of $f$ itself, so

$$0 = \left(c + bt + at^2\right)' \tag{1.9}$$

$$0 = b + 2at \tag{1.10}$$

$$t_{\min} = \frac{-b}{2a} \tag{1.11}$$

### 2. Derivation of $r_{min}$, the minimum range, or point of closest approach for the case described in 1).

Having derived $t_{\min} = \frac{-b}{2a}$ we merely substitute $t_{min}$ for $t$ in the general equation of 1.1:

$$r_{\min} = \sqrt{c + bt_{\min} + at_{\min}^2} \tag{2.1}$$

$$= \sqrt{c + b\frac{-b}{2a} + a\left(\frac{-b}{2a}\right)^2} \tag{2.2}$$

$$= \sqrt{c - \frac{b^2}{2a} + \frac{b^2}{4a}} \tag{2.3}$$

$$= \sqrt{\frac{4ac - 2b^2 + b^2}{4a}} \tag{2.4}$$

$$= \sqrt{\frac{4ac - b^2}{4a}} \tag{2.5}$$

### 3. The general case where $r \neq 0$

In the general case where $r \neq 0$, to find the times $t_r$ when the intruder will enter and exit the protected zone we must follow the logic of 1), but begin with the following modification of Equation 1.4:

$$r = \sqrt{c + bt_r + at_r^2} \tag{3.1}$$

and solve for $r$. Squaring both sides, we get:

$$r^2 = c + bt_r + at_r^2 \tag{3.2}$$

which is conveniently set up to allow us to use the quadratic formula by setting the left-hand side of 3.2 to zero:

$$0 = \left(c - r^2\right) + bt_r + at_r^2 \tag{3.3}$$

$$0 = \left(c_2\right) + bt_r + at_r^2 \tag{3.4}$$

where $c_2 = \left(c - r^2\right)$, (3.5)

For the standard quadratic form $0 = ax^2 + bx + c$, the quadratic roots here will be

$$t_r = \frac{-b \pm \sqrt{b^2 - 4ac_2}}{2a} \tag{3.6}$$

The smaller value of which will represent the time of entry into the protected zone and the larger value of which will represent the time of exit from the protected zone.

## Appendix 2: What the SCAMP program is and does

SCAMP is a Mathematica program that allows the user to visualize and calculate the congestive effect of defined air traffic on an ownship flying in a defined airspace. It first allows you to generate traffic for important cases, e.g. diverging traffic, parallel paths, and X-crossings. Second, you can generate a map view of this traffic. This map helps you jockey aircraft into the exact places, with the exact headings, speeds, and altitudes you want in order to test critical situations. Third, SCAMP lets you define a maneuver space -- a 3-D representational space consisting of a heading dimension, a speed dimension, and an altitude dimension. The maneuver space represents all the maneuver combinations an aircraft (aircraft) could make, given some specified amount of time. Fourth, SCAMP calculates the conflict regions, which are cells - combinations of heading, speed, and altitude - in the maneuver space. These cells have been calculated by the program's conflict probe to result in a violation, were the ownship to adopt that vector.

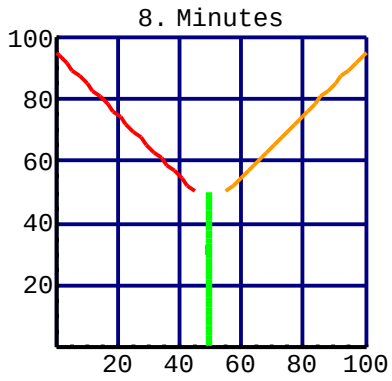
Finally, SCAMP calculates a metric based on the maneuver space and the conflict regions. This metric is explained more fully in the Define SCAMP Metric section.

| | | | |
|---|---|---|---|
| SPD (Speed) | nm./ hr. (groundspeed) | CLMRT | climb rate, ft / min |
| HDG (Heading) | degrees, math coordinates | A | is HDG angle in radians |
| X,Y | initial positions, degrees | Vx, Vy | are velocities, nm / min |
| Z | initial altitude, ft. | Vz | is also in nm / min |

```
SECTORTOP = 41000; (*Ceiling altitude of your sector*)
SECTORBTM=31000;  (*Floor altitude of sector *)
MIDDLE=SECTORBTM + (SECTORTOP-SECTORBTM)/2;
```

### Example 1: Everybody diverging

This shows up best when viewed head-on, View Point-> {0,2,0}. You can see the "hole" in the center.




**Map view of the traffic situation in example 1.**

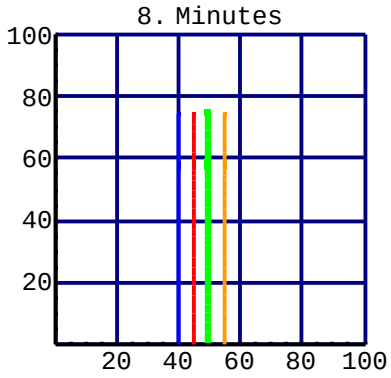
```
 MAXPLANES=5; (*MAKE THIS AT LEAST AS BIG AS THE # OF PLANES!!!*)
NumPlanes=0; (*Counter, to keep track of the actual no. of aircraft*)
Plane = Table[0,{ MAXPLANES}];
Spd=Hdg=XO=YO=ZO=Clmrt=\[Alpha]=Vx=Vy=Vz=Plane;(*"that's "XOh", not
"zero"*)
Ownship = 1;                (* 1 shows aircraft, 0 suppresses it *)
If[Ownship==1, Plane[[1]] = 1;        NumPlanes++;
    Spd[[1]]=   560.0;
    Hdg[[1]]= 270.0;
    XO[[1]]    =    050.0;       (*XO stands for "X-original"*)
    YO[[1]]    =    050.0;
    ZO[[1]]    =    MIDDLE;
    Clmrt[[1]]=    000.0;   (*Rate of climb, in ft/min*)
    \[Alpha][[1]]= (Hdg[[1]]*Pi)/180.0; (*alpha is just converted to radians*)
    Vx[[1]]=N[(Spd[[1]]/60.)*Cos[\[Alpha][[1]] ]];
```

```
    Vy[[1]]=N[(Spd[[1]]/60.)*Sin[\[Alpha][[1]] ]];
    Vz[[1]]=N[Clmrt[[1]]/6080.2]
   ];

Plane2 = 1;
If[Plane2==1, Plane[[2]] = 1;      NumPlanes++;
    Spd[[2]]=   560.0;
    Hdg[[2]]=  045.0;
    XO[[2]]    =    055.1;
    YO[[2]]    =   050.0;
    ZO[[2]]    =   MIDDLE;
    Clmrt[[2]]=    000.0;
    \[Alpha][[2]]=(Hdg[[2]]*Pi)/180.0;
    Vx[[2]]=N[(Spd[[2]]/60.)*Cos[\[Alpha][[2]] ]];
    Vy[[2]]=N[(Spd[[2]]/60.)*Sin[\[Alpha][[2]] ]];
    Vz[[2]]=N[Clmrt[[2]]/6080.2]
];

Plane3 = 1;
If[Plane3==1, Plane[[3]] = 1;      NumPlanes++;
    Spd[[3]]=   560.0;
    Hdg[[3]]=  135.0;
    XO[[3]]    =    044.9;
    YO[[3]]    =   050.0;
    ZO[[3]] =   MIDDLE;
    Clmrt[[3]]=    000.0;
    \[Alpha][[3]]=(Hdg[[3]]*Pi)/180.0;
    Vx[[3]]=N[(Spd[[3]]/60.)*Cos[\[Alpha][[3]] ]];
    Vy[[3]]=N[(Spd[[3]]/60.)*Sin[\[Alpha][[3]] ]];
    Vz[[3]]=N[Clmrt[[3]]/6080.2]
];

Plane4 = 0;
If[Plane4==1, Plane[[4]] = 1;      NumPlanes++;
    Spd[[4]]=  560.0;
    Hdg[[4]]=  90.0;
    XO[[4]]    =    039.8;
    YO[[4]]    =   000.0;
    ZO[[4]] =   MIDDLE-1000;
    Clmrt[[4]]=    100.0;
    \[Alpha][[4]]=(Hdg[[4]]*Pi)/180.0;
```

```
    Vx[[4]]=N[(Spd[[4]]/60.)*Cos[\[Alpha][[4]]]];
    Vy[[4]]=N[(Spd[[4]]/60.)*Sin[\[Alpha][[4]]]];
    Vz[[4]]=N[Clmrt[[4]]/6080.2]
];
```

### Example 2: Parallel paths with 5.1 mile offset

This shows up best when viewed head-on, View Point-> {0,2,0}. You can see the "hole" in the center.



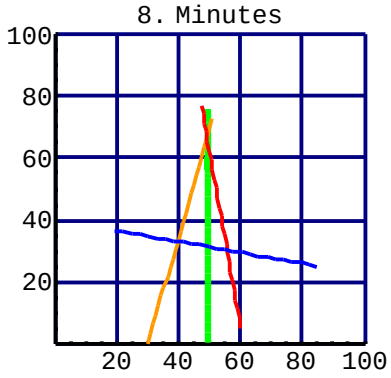

**Map view of the traffic situation in example 2.**

```
MAXPLANES=5; (*MAKE THIS AT LEAST AS BIG AS THE # OF PLANES!!!*)
NumPlanes=0; (*Counter, to keep track of the actual no. of aircraft*)
Plane = Table[0,{ MAXPLANES}];
Spd=Hdg=XO=YO=ZO=Clmrt=\[Alpha]=Vx=Vy=Vz=Plane;(*"that's "XOh", not
"zero"*)
Ownship = 1;                    (* 1 shows aircraft, 0 suppresses it *)
If[Ownship==1, Plane[[1]] = 1;          NumPlanes++;
    Spd[[1]]=   560.0;
    Hdg[[1]]=   090.0;
    XO[[1]]    =    050.0;       (*XO stands for "X-original"*)
    YO[[1]]    =    000.0;
    ZO[[1]]    =    MIDDLE;
    Clmrt[[1]]=    000.0;   (*Rate of climb, in ft/min*)
    \[Alpha][[1]]= (Hdg[[1]]*Pi)/180.0; (*alpha is just converted to radians*)
```

```
    Vx[[1]]=N[(Spd[[1]]/60.)*Cos[\[Alpha][[1]] ]];
    Vy[[1]]=N[(Spd[[1]]/60.)*Sin[\[Alpha][[1]] ]];
    Vz[[1]]=N[Clmrt[[1]]/6080.2]
  ];

Plane2 = 1;
If[Plane2==1, Plane[[2]] = 1;      NumPlanes++;
    Spd[[2]]=   560.0;
    Hdg[[2]]=  090.0;
    XO[[2]]    =    055.1;
    YO[[2]]    =   000.0;
    ZO[[2]]    =   MIDDLE;
    Clmrt[[2]]=    000.0;
    \[Alpha][[2]]=(Hdg[[2]]*Pi)/180.0;
    Vx[[2]]=N[(Spd[[2]]/60.)*Cos[\[Alpha][[2]] ]];
    Vy[[2]]=N[(Spd[[2]]/60.)*Sin[\[Alpha][[2]] ]];
    Vz[[2]]=N[Clmrt[[2]]/6080.2]
];

Plane3 = 1;
If[Plane3==1, Plane[[3]] = 1;      NumPlanes++;
    Spd[[3]]=   560.0;
    Hdg[[3]]=  090.0;
    XO[[3]]    =    044.9;
    YO[[3]]    =   000.0;
    ZO[[3]] =   MIDDLE;
    Clmrt[[3]]=    000.0;
    \[Alpha][[3]]=(Hdg[[3]]*Pi)/180.0;
    Vx[[3]]=N[(Spd[[3]]/60.)*Cos[\[Alpha][[3]] ]];
    Vy[[3]]=N[(Spd[[3]]/60.)*Sin[\[Alpha][[3]] ]];
    Vz[[3]]=N[Clmrt[[3]]/6080.2]
];

Plane4 = 1;
If[Plane4==1, Plane[[4]] = 1;      NumPlanes++;
    Spd[[4]]=  560.0;
    Hdg[[4]]=  90.0;
    XO[[4]]    =    039.8;
    YO[[4]]    =   000.0;
    ZO[[4]] =   MIDDLE;
    Clmrt[[4]]=    000.0;
```

```
    \[Alpha][[4]]=(Hdg[[4]]*Pi)/180.0;
    Vx[[4]]=N[(Spd[[4]]/60.)*Cos[\[Alpha][[4]]]];
    Vy[[4]]=N[(Spd[[4]]/60.)*Sin[\[Alpha][[4]]]];
    Vz[[4]]=N[Clmrt[[4]]/6080.2]
];
```

### Example 3: Intruders @ X-crossing




**Map view of the traffic situation in example 3.**

```
MAXPLANES=5; (*MAKE THIS AT LEAST AS BIG AS THE # OF PLANES!!!*)
NumPlanes=0; (*Counter, to keep track of the actual no. of aircraft*)
Plane = Table[0,{ MAXPLANES}];
Spd=Hdg=XO=YO=ZO=Clmrt=\[Alpha]=Vx=Vy=Vz=Plane;(*"that's "XOh", not
"zero"*)
Ownship = 1;                    (* 1 shows aircraft, 0 suppresses it *)
If[Ownship==1, Plane[[1]] = 1;          NumPlanes++;
    Spd[[1]]=   560.0;
    Hdg[[1]]=   090.0;
    XO[[1]]    =    050.0;       (*XO stands for "X-original"*)
    YO[[1]]    =    000.0;
    ZO[[1]]    =    MIDDLE;
    Clmrt[[1]]=    000.0;
    \[Alpha][[1]]= (Hdg[[1]]*Pi)/180.0; (*alpha is just converted to radians*)
    Vx[[1]]=N[(Spd[[1]]/60.)*Cos[\[Alpha][[1]] ]];
    Vy[[1]]=N[(Spd[[1]]/60.)*Sin[\[Alpha][[1]] ]];
    Vz[[1]]=N[Clmrt[[1]]/6080.2]
  ];
```

```
Plane2 = 1;
If[Plane2==1, Plane[[2]] = 1; NumPlanes++;
    Spd[[2]]=   565.0;
    Hdg[[2]]=  74.0;
    XO[[2]]    =    030.0;
    YO[[2]]    =   000.0;
    ZO[[2]]    =   MIDDLE;
    Clmrt[[2]]=    000.0;
    \[Alpha][[2]]=(Hdg[[2]]*Pi)/180.0;
    Vx[[2]]=N[(Spd[[2]]/60.)*Cos[\[Alpha][[2]] ]];
    Vy[[2]]=N[(Spd[[2]]/60.)*Sin[\[Alpha][[2]] ]];
    Vz[[2]]=N[Clmrt[[2]]/6080.2]
];

Plane3 = 1;
If[Plane3==1, Plane[[3]] = 1; NumPlanes++;
    Spd[[3]]=   550.0;
    Hdg[[3]]=  100.0;
    XO[[3]]    =    060.0;
    YO[[3]]    =   005.0;
    ZO[[3]] =   MIDDLE;
    Clmrt[[3]]=    000.0;
    \[Alpha][[3]]=(Hdg[[3]]*Pi)/180.0;
    Vx[[3]]=N[(Spd[[3]]/60.)*Cos[\[Alpha][[3]] ]];
    Vy[[3]]=N[(Spd[[3]]/60.)*Sin[\[Alpha][[3]] ]];
    Vz[[3]]=N[Clmrt[[3]]/6080.2]
];

Plane4 = 1;
If[Plane4==1, Plane[[4]] = 1; NumPlanes++;
    Spd[[4]]=  500.0;
    Hdg[[4]]=  170.0;
    XO[[4]]    =    085.0;
    YO[[4]]    =   025.0;
    ZO[[4]] =   MIDDLE;
    Clmrt[[4]]=    000.0;
    \[Alpha][[4]]=(Hdg[[4]]*Pi)/180.0;
    Vx[[4]]=N[(Spd[[4]]/60.)*Cos[\[Alpha][[4]]]];
    Vy[[4]]=N[(Spd[[4]]/60.)*Sin[\[Alpha][[4]]]];
    Vz[[4]]=N[Clmrt[[4]]/6080.2]
```

```
];

START     = 0.01;        FINISH = 8.0;     (*time in minutes*)
(*Below is a plot of the ships and their relative headings *)
Avion={};
If[ Plane[[1]] == 1, G1=ParametricPlot[{XO[[1]]+t*Vx[[1]],
     YO[[1]]+t*Vy[[1]] },{t,START,FINISH},
     DisplayFunction->Identity,
     PlotStyle->{{RGBColor[0,1,0], (*Green*) Thickness[0.02]}}];
     AppendTo[Avion,G1], AppendTo[Avion,{}]
   ];
If[ Plane[[2]] == 1, G2=ParametricPlot[{XO[[2]]+t*Vx[[2]],
     YO[[2]]+t*Vy[[2]] },{t,START,FINISH},
     DisplayFunction->Identity,
     PlotStyle->{{RGBColor[1,0.6,0],(*Orange*) Thickness[0.01]}}];
     AppendTo[Avion,G2], AppendTo[Avion,{}]
   ];
If[ Plane[[3]] == 1, G3=ParametricPlot[{XO[[3]]+t*Vx[[3]],
     YO[[3]]+t*Vy[[3]] },{t,START,FINISH},
     DisplayFunction->Identity,
     PlotStyle->{{RGBColor[1,0,0], (*Red*) Thickness[0.01]}}];
     AppendTo[Avion,G3], AppendTo[Avion,{}]
   ];
If[ Plane[[4]] == 1, G4=ParametricPlot[{XO[[4]]+t*Vx[[4]],
     YO[[4]]+t*Vy[[4]] },{t,START,FINISH},
     DisplayFunction->Identity,
     PlotStyle->{{RGBColor[0,0,1], (*Blue*) Thickness[0.01]}}];
     AppendTo[Avion,G4], AppendTo[Avion,{}]
   ];

Show[Avion,
        PlotRange->{{0,100},{0,100}},GridLines->Automatic,
        PlotLabel-> FINISH "Minutes",AspectRatio->1/1,
        DisplayFunction->$DisplayFunction] ;
```

## Definition of the SCAMP Metric

### (V1=Ramp function, V2=gaussians x Tc)

This is where we define the advanced metric of congestion. Each cell in the maneuver space is defined as being hstep X sstep X zstep wide. If you have a conflict cell, this weights it according to the integral of a Gaussian function on that dimension. The term m (mu) shifts the function left and right, so we just set it = zero.

The standard deviations s (sigma) of the Gaussians are adjustable, one per dimension). This allows us to factor-weight each component separately, one per dimension of hdg/spd/alt.

This is a considered attempt to capture the idea of "If it's right in my face, it's more important , because it means I HAVE to maneuver. If it's off to the side of me, it still contributes to 'congestion', but if the deviation I'd have to make is huge, I'll probably go with a solution in another dimension, so the weighting factor for overall congestion on that first dimension won't increase linearly. It'll still increase, if there's a lot of conflict in that dimension--but the amount it changes will matter less and less, the farther out I'd have to deviate"

The final Gaussian-component approach is cool, because, not only does it model how pilots and ATC think, but it accounts for variance due to 3 factors--heading, speed, and altitude constraints to maneuver. Each

factor has a tunable parameter and we can ultimately thus tune the global composite function--and test its validity --by curve-fitting to expert ratings re congestion in actual traffic situations.

The function ScampSez (below) is only called when the conflict probe determines that there'd be a conflict at that particular combination of heading/speed/altitude.

```
V1 = 1;  V2 = 2;

ScampSez[mode_, whichplane_, hstp_, sstp_, zstp_, hd_, sp_, alt_,
   Tmin_] :=

 Module[{CongestionIndex, hwt, swt, zwt, \[Sigma]h, \[Sigma]s, \[Sigma]a, x,
    y, z, Hdif, Sdif, Adif},

    If[mode == V2,
          \[Sigma]h =
     hstp;(*Specify std. deviations of gaussians*)
          \[Sigma]s =
     sstp;
       \[Sigma]a = zstp;
       a = 3.27;(*Specify parameters of sigmoid*)
       b = 0;
       cc = -.99;
       w = 3.14;
          gh = (
       1/(\[Sigma]h*
          Sqrt[2*Pi] ) ) * (E^( -(x^2)/(2*(\[Sigma]h^2) ) ) );
       gs = ( 1/(\[Sigma]s*Sqrt[2*Pi] ) ) * (E^( -(y^2)/(2*(\[Sigma]s^2) ) ) );
       ga = ( 1/(\[Sigma]a*Sqrt[2*Pi] ) ) * (E^( -(z^2)/(2*(\[Sigma]a^2) ) ) );
       sig = (  (1 - b)/(1 + (a^(w*(Tmin + cc))) )  )   + b;

       Hdif = Abs[Hdg[[whichplane]] - hd];
       Sdif = Abs[Spd[[whichplane]] - sp];
       Adif = Abs[ZO[[whichplane]] - alt];
```

```
        RawCongestionIndex = N[Integrate[gh*gs*ga ,
            {x, Hdif - (0.5*hstp), Hdif + (0.5*hstp)},
            {y, Sdif - (0.5*sstp), Sdif + (0.5*sstp)},
            {z, Adif - (0.5*zstp), Adif + (0.5*zstp)}]];

            Return[N[RawCongestionIndex]];
          ]   (*End If[mode == V2]*)

   ];  (*End ScampSez module*)
```

**Calculate & Show Maneuver Space, Conflict Region, & SCAMP Results**

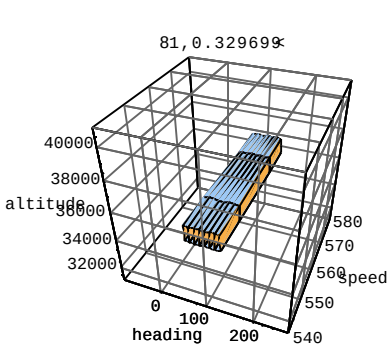
This calculates a component index of congestion, airplane by airplane

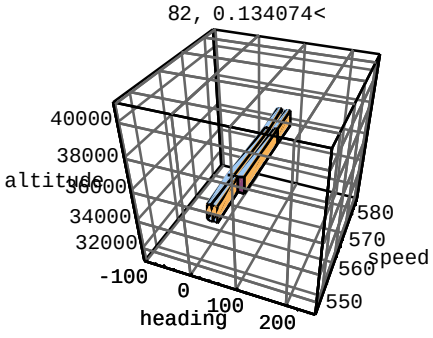
```
hstep = 10.;      (*Heading granularity (cell size),  degrees*)

zstep = 1000.;  (*Altitude granularity (cell size),  feet*)

sstep = 20.;      (*Speed granularity (cell size),  knots*)

NumSsteps = 1; (* +/- the no. of ssteps to examine for conflict*)

LookAhead = 8; (*Conflict probe lookahead,  minutes*)

MODE = V2; (*Which version of SCAMP metric to use*)

BigListOCubes = {};
Speeds = Table[{0, 0}, { NumPlanes}];       (*1st ele = Min, 2nd ele = Max*)

AllCubesForOnePlane = Table[{}, { NumPlanes}];
RunnyNose =
   Table[0, { NumPlanes}, {
      NumPlanes}]; (*RunnyNose holds values of congestion for each conflict*)
\
MinZed =   SECTORBTM;        MaxZed =  SECTORTOP;
Print["The following on-course conflicts existed:"];
(*Below, we have to set up a vector to tell us which, if any,
  cubes have already  been conflictual,
```

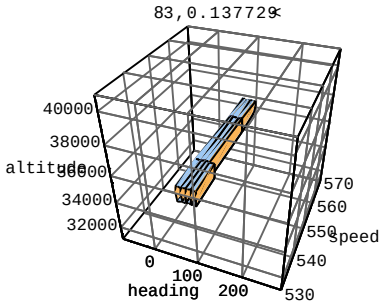
```
 so that we don't count them more than once in the congestion analysis*)
For[
 i = 1, i < NumPlanes, i++,
 If[Plane[[i]] == 1, (*Meaning this aircraft is active in this scenario*)

    MinHdg = Hdg[[i]] - 180;  MaxHdg = Hdg[[i]] + 180;(*"\!\(\*
StyleBox[\"Ownship\",\nCellMargins->{{46, Inherited}, {Inherited, \
Inherited}},\nGroupPageBreakWithin->Automatic,\nAspectRatioFixed->True,\n\
FontFamily->\"Helvetica\"]\)\!\(\*
StyleBox[\"\\\"\",\nCellMargins->{{46, Inherited}, {Inherited, Inherited}},\n\
GroupPageBreakWithin->Automatic,\nAspectRatioFixed->True,\n\
FontFamily->\"Helvetica\"]\) values*)

   MinSpd = Spd[[i]] - (NumSsteps*sstep);
   MaxSpd = Spd[[i]] + (NumSsteps*sstep);
            Speeds[[i, 1]] = MinSpd;     Speeds[[i, 2]] = MaxSpd;
   UsedCubes = Table[0, {Round[(MaxHdg - MinHdg)/hstep]},
      {1 +
         Round[(MaxSpd - MinSpd)/sstep]}, {Round[(MaxZed - MinZed)/zstep]}
          ];  (*End If[Plane[[i]]*)
  For[j = i + 1, j ≤ NumPlanes, j++,
    If[Plane[[j]] == 1,
     Congestion = 0.0;
     Block[{\[Alpha]0, Vx0, Vy0, k1, k2, k3, k4, k5, k6, k7, k8, k9},
      ListOCubes = {};
      (*Below, heading is < to avoid duplicating the final value,
        speed is <=, to cover the full range, and Z is <, to avoid
          climbing past ceiling of the sector*)

       For[  Hd = MinHdg; ii = 1,  Hd < MaxHdg,   Hd += hstep; ii++,
         For[  Sp = MinSpd; jj = 1, Sp <= MaxSpd, Sp += sstep; jj++,
           For[Z = MinZed; kk = 1, Z < MaxZed, Z += zstep; kk++,
            \[Alpha]0 = (Hd*Pi)/180.0;(*\[Alpha] - zero, at time zero*)


             Vx0 = N[(Sp/60.0) * Cos[\[Alpha]0]];   (* nm / min *)
             Vy0 = N[(Sp/60.0) * Sin[\[Alpha]0]];
              k1 = Vx0 - Vx[[j]];   k2 = Vy0 - Vy[[j]];
             k3 = Vz[[i]] - Vz[[j]]; k4 = XO[[i]] - XO[[j]];
             k5 = YO[[i]] - YO[[j]];
             k6 = (ZO[[i]] - ZO[[j]])/6080.2;
```

```
k7 = k1^2   + k2^2  + k3^2;
k8 = k1*k4 + k2*k5 + k3*k6;
k9 = k4^2   + k5^2  + k6^2;
If[Abs[k7] < 0.000001, k7 = 0.000001];
tmin = -k8/k7;        (* Tc, in min.*)

temp = k7*(tmin^2) + 2*k8*tmin + k9;

If[temp > 0, rmin = Sqrt[temp],  (* Rmin, in nm.*)

          rmin = 0.000001];

ZSep = Abs[(Z + Clmrt[[i]]*tmin) - (ZO[[j]] +
         Clmrt[[j]]*tmin)] ;

If[ (tmin >= 0.000001 && rmin <= 5.000001  &&
     tmin < LookAhead
     && ZSep < 1000),
  (*Print["       rmin: ", N[rmin, 3], "  tmin: ",
      N[tmin, 3], " min."];*)

  If[UsedCubes[[ii, jj, kk]] == 0,
   If[Hd == Hdg[[i]] && Sp == Spd[[i]],
    Print["i: ", i, "   j: ", j]];

   Congestion +=
    ScampSez[MODE, i, hstep, sstep, zstep, Hd, Sp, Z,
     tmin];
   AppendTo[ListOCubes,

    Cuboid[{Hd - .5hstep, Sp - .5sstep,
       Z - .5zstep}, {Hd + .5hstep, Sp + .5sstep,
       Z + .5zstep}]];
   AppendTo[AllCubesForOnePlane[[i]],

    Cuboid[{Hd - .5hstep, Sp - .5sstep,
       Z - .5zstep}, {Hd + .5hstep, Sp + .5sstep,
       Z + .5zstep}]];
   UsedCubes[[ii, jj, kk]] = 1
   ]
  ](*End If[tmin >=*)
```

```
        ]]] ;        (*End For[Hd, Sp,
              Z] loops*)
         AppendTo[BigListOCubes, ListOCubes];
         If[MODE == V1, RunnyNose[[i, j]] = Congestion/TotalCongestionV1];
         If[MODE == V2, RunnyNose[[i, j]] = Congestion]
         ]   (*End Block*)
       ](*End If[Plane[[j]] =*)
     ](*End For[
        j =]*)
   ](*End If[Plane[[i]] =*)
 ];(*End For[i =]*)

(*The following displays the maneuver space and conflict regions for
     aircraft which SCAMP has just examined.  The last number in the
        bracketed list{  } is the value of the SCAMP metric for that \
conflict.  These
        pix are the result of Example 3 *)
For[i = 1, i < NumPlanes, i++,
  If[AllCubesForOnePlane[[i]] ≠ {},
    TotalStuffiness = 0;
    For[j = 1, j <= NumPlanes, j++, TotalStuffiness += RunnyNose[[i, j]]];
    Show[Graphics3D[AllCubesForOnePlane[[i]]],
      View Point -> {.8, -2, 1.5}, (* {0, 2, 0}, *)

      PlotLabel -> {i, TotalStuffiness},
      BoxRatios -> {1, 1, 1}, Axes -> True,(*RenderAll -> False,*)

      AxesLabel -> {"heading", "speed", "altitude"}, FaceGrids -> All,
      PlotRange -> {{Hdg[[i]] - 180, Hdg[[i]] + 180},
      {Speeds[[i, 1]], Speeds[[i, 2]]}, {SECTORBTM, SECTORTOP}}]
    ]];
```

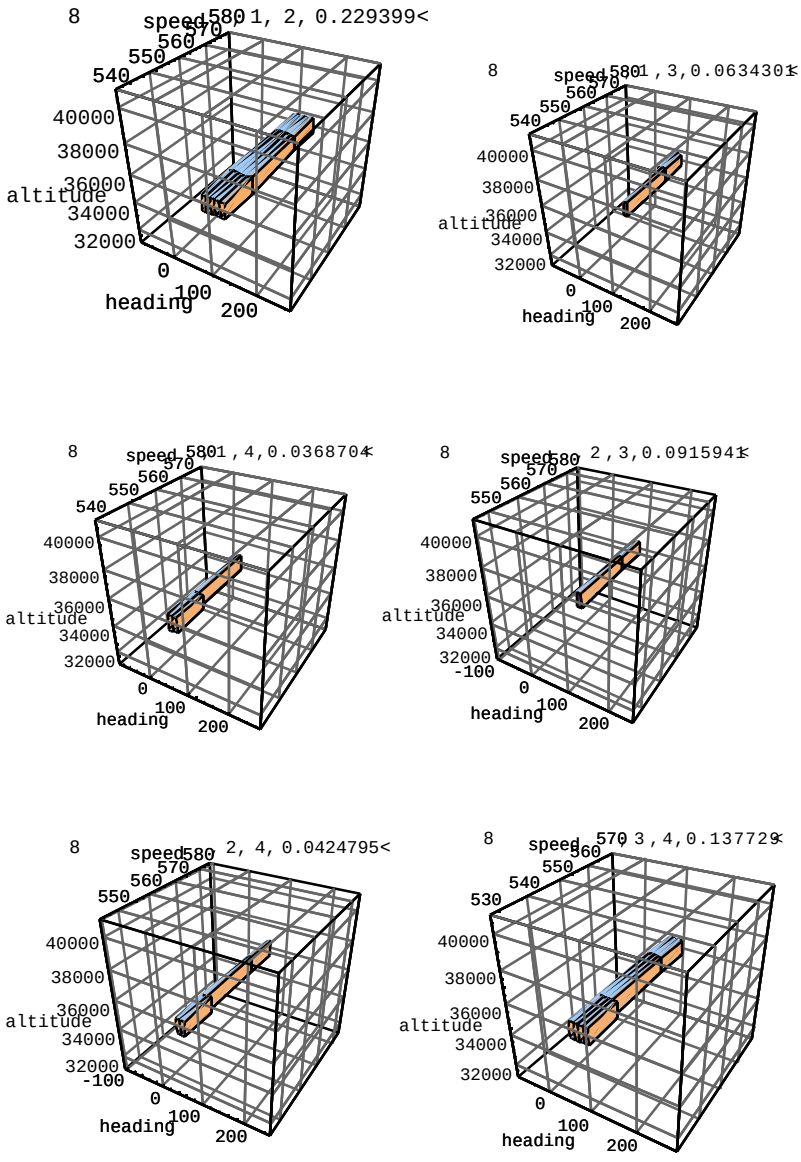



```
(* Below we calculate the maneuver space and conflict region for each aircraft pair.
Again, the metric
    is the last # in the bracketed list *)
index = 1;
For[i = 1, i < NumPlanes, i++,
 If[Plane[[i]] == 1,
  For[j = i + 1, j ≤ NumPlanes, j++,
   If[Plane[[j]] == 1,
    NumConflicts = Length[BigListOCubes[[index]]];
    Show[Graphics3D[BigListOCubes[[index]]],
     View Point -> {1.2, -2, 1.5}(*{0, 2, 0}*),
     PlotLabel -> {"          ", i, j, RunnyNose[[i, j]]},
     BoxRatios -> {1, 1, 1}, Axes -> True,(*RenderAll -> False,*)

       AxesLabel -> {"heading", "speed", "altitude"}, FaceGrids -> All,
     PlotRange -> {{Hdg[[i]] - 180, Hdg[[i]] + 180},
```

```
        {Speeds[[i, 1]], Speeds[[i, 2]]}, {SECTORBTM, SECTORTOP}}];
    index++
    ]]]];
```



```
(*Total sector congestion models.  This last part is purely experimental*)

(*This section hasn't been fully implemented yet*)
(*Model 1 -- Simple \
pairwise average*)
TotalIndex = 0;  TotalPlanes = 0;
For[i = 1, i < NumPlanes, i++,
   If[Plane[[i]] == 1,
     For[j = i + 1, j ≤ NumPlanes, j++,
       If[Plane[[j]] == 1,
         Print["Contribution of: ", i, " x ", j, " : ", RunnyNose[[i, j]]];
         TotalPlanes++;
         TotalIndex += RunnyNose[[i, j]];
         ]]]];
Print["Total index, Model 1: ", TotalIndex/TotalPlanes];
(*Model 2* --
     Thresholded pairwise average*)
TotalIndex = 0;  TotalPlanes = 0;
\[Theta] = .12;
For[i = 1, i < NumPlanes, i++,
   If[Plane[[i]] == 1,
     For[j = i + 1, j ≤ NumPlanes, j++,
       If[Plane[[j]] == 1,
         If[RunnyNose[[i, j]] > \[Theta],

            Print["Contribution of: ", i, " x ", j, " : ",
              RunnyNose[[i, j]]];
            TotalPlanes++;
            TotalIndex += RunnyNose[[i, j]];
         ]]]];
Print["Total index, Model 2: ", TotalIndex/TotalPlanes];
(*Model 3 -- Display max value only*)
Biggest = 0;
For[i = 1, i < NumPlanes, i++,
   If[Plane[[i]] == 1,
     For[j = i + 1, j ≤ NumPlanes, j++,
       If[Plane[[j]] == 1,
         If[RunnyNose[[i, j]] > Biggest,
            Biggest = RunnyNose[[i, j]]];
         ]]]];
Print["Max Index, Model 3: ", Biggest];
```