ON OPTIMISING FAC(M) COUNTER MISSILE TACTICS

-

A DYNAMIC SIMULATION MODEL TO OPTIMISE SOFT KILL TACTICS EMPLOYED BY A GENERIC FAST ATTACK CRAFT AGAINST A GENERIC SURFACE-TO-SURFACE, FIRE-AND-FORGET MISSILE

by

GERHARD NIEUWOUDT ENGELBRECHT

submitted in part fulfilment of the requirements for the degree of

MASTER OF SCIENCE

in the subject

OPERATIONS RESEARCH

at the

UNIVERSITY OF SOUTH AFRICA

Supervisor: Prof CJ Swanepoel, WDS

November 1997

Summary

ON OPTIMISING FAC(M) COUNTER MISSILE TACTICS

The aim of this dissertation is to show how counter missile tactics for a fast attack craft armed with missiles [FAC(M)] against a surface-to-surface, fire-and-forget missile [SSM] can be optimised. As a result the ship and missile will be modelled as generic concepts while the environment will be a chosen area of operations. The applicable methodology is to simulate the ship, missile and environment as well as the interactions between them. At the same time, the ship will be carrying out combinations of five separate missile counter measures.

The methodology is then to build a dynamic simulation model to optimise soft kill tactics by a generic FAC(M) against a generic SSM in the chosen environment and evaluate the outcome of the simulation by viewing the experiment as a 2^5 factorial design and to analyse it accordingly.

KEYWORDS

Anti-missile tactics
Fast attack craft
Maritime environment
Missile counter measures
Naval warfare
Surface warfare
Surface-to-surface missile

Preface

The main reason for embarking on this venture is to show how the missile counter-measure problem can be resolved by scientific means. The driver of this idea must have been the very words of Pepys and Churchill as quoted overleaf. The beginnings of the process were, in general, the Weapons Officer's Qualifying Course for S.A. Naval Strike Craft which I attended during 1979 and, in particular, the lectures on statistics during that course by the late Dr. J.C. Visser from the Institute for Maritime Technology. The continuation of the idea was fueled by the late Cdr Dick Benson whose enthusiasm for gunnery and missilery was an inspiration to aspire to this study. The rest was many years of study under the competent guidance of my Unisa lecturers.

In particular, I wish to thank my study leader, Professor C.J. Swanepoel, WDS, for his valuable contributions which enabled me to deliver a product which is academically well founded and Ms Zita Nawn for editing the script to ensure the proper use of the English language.

Captain G.N. Engelbrecht, S.A. Navy.
Centurion
July 1997

That Integrity *and general (but unpractic'd)* Knowledge,
are not alone Sufficient to conduct and Support a Navy *so,*
as to prevent its Declenſion *into a State little less unhappy,*
than the worst that can befall it under the want of both.

Samual Pepys
Memoires Relating to the State of the
Royal Navy of England for Ten Years,
Determin'd 1688
London 1690

It is by devising new weapons,
and above all by scientific leadership,
that we shall best cope with the
enemy's superior strength.

Winston Churchill
Memorandum for the War Cabinet
3 September 1940

Dedicated to

Rear Admiral Johan F. Retief SD, SM, MMM
whose intervention made it possible

and to

Meryl, Ewald and Hannelie Engelbrecht
for their loving support and patience.

List of Contents

List of Figures

List of Tables

Chapter 1

HISTORICAL OVERVIEW

1.1 THE RUN UP TO MISSILE WARFARE AT SEA [1956 - 1967]

On 15 of July 1956, in Cardiff Docks, the Royal Navy transferred a Z-class destroyer, H.M.S. ZEALOUS, to the Israeli Navy. With a full load displacement of 2 555 tons, a maximum speed of 31 knots, she was armed with four 4.5" dual purpose, that is Anti Aircraft (AA) and Surface (SU) guns, six 40 mm AA guns, 4 depth charge throwers and eight 21" torpedo tubes.[1]

Her new owners renamed her INS ELATH and stationed her, as part of their naval defence strategy for the Mediterranean, in the port of Tel Aviv. Although not the most advance ship of her kind, in the Middle East context, INS ELATH was indeed a formidable weapon.

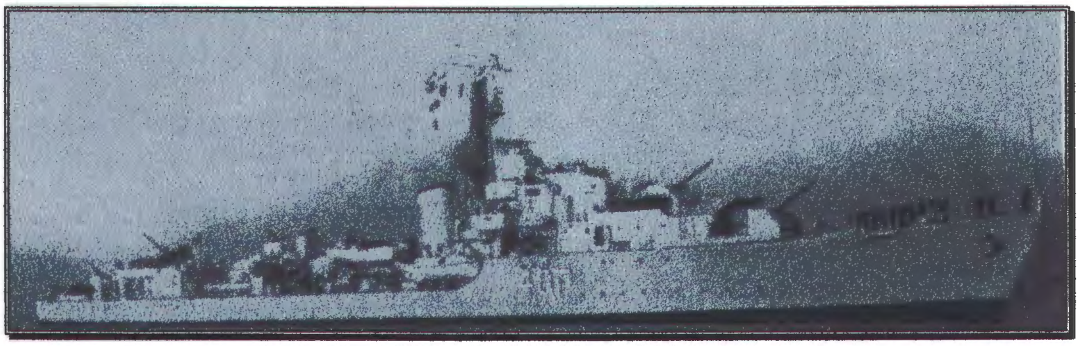


Figure 1.1 : INS ELATH

During the early 1950's, the Soviet Navy was involved in developing another new weapon, a fire-and-forget surface-to-surface missile (SSM). It was named the Styx shipborne SSM and code-named by NATO as SSN-2. The general configuration of the SSN-2 was that of a small aircraft, with a delta planform wing and a triple tail surface arrangement. A jettisonable booster rocket was used for the launch and acceleration phases after which an internal motor maintains a cruising speed of about Mach 0,9. Its range[2] was estimated at approximately 20 nautical miles (nm).

At the same time the Soviet Navy was developing several classes of new Missile Patrol Boats, later classed as fast attack craft - missile [FAC(M)]. One type was named the OSA-class FAC(M), the other the KOMAR-class. The new SSN-2 missiles were fitted to them. The OSAs displaced 200 tons at full load, were about 40 metres long, could run at a maximum speed of 35 knots and were armed with four 25

[1] *Jane's Fighting Ships 1967/68*, edited by Capt J.E. Moore. London : Jane's Yearbooks, [1967], p. 146
[2] *Jane's Weapon Systems 1980/81* edited by R.T. Pretty. London : Jane's Yearbooks,[1980], p. 61

mm AA cannons and four SSN-2 missiles. By 1966 the Soviet Navy had 50 OSAs in her inventory. The KOMARs displaced only 100 tons at full load, were about 27 metres long, could run at a maximum speed of 40 knots and were armed with two 25 mm AA cannons and two SSN-2 missiles. By 1966 the Soviet Navy also had 50 KOMARs in their inventory.

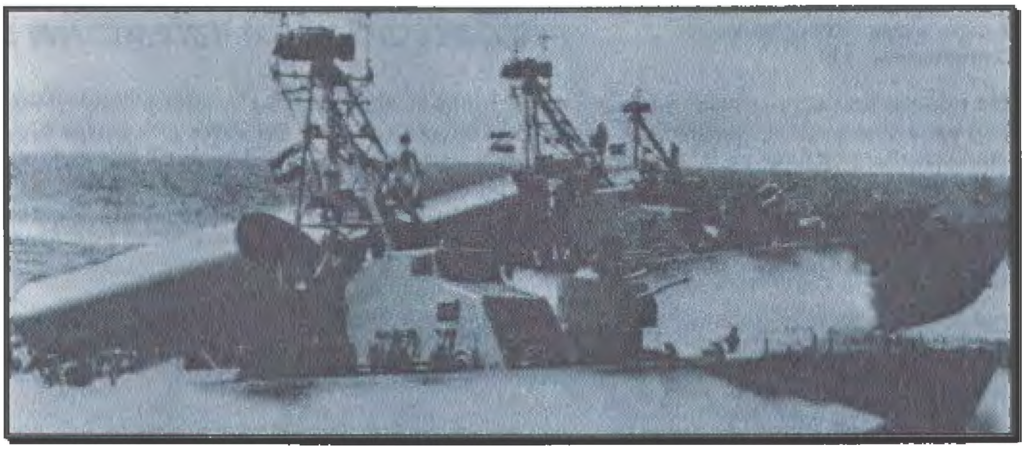


Figure 1.2 : KOMAR-Class FAC(M) in the Egyptian Inventory.

The USSR also were exporting these ships to allies and "friends" all over the globe. By 1966 the Egyptian Navy owned ten OSA-class and eight KOMAR-class FAC(M) as part of her inventory.

By the mid 1960's several navies embarked on building SSMs. These, yet untested, weapons caused much speculation and often parallels with the torpedo development of the first decade of the twentieth century were drawn. Some believed that these new weapons would fail. The hit probability of the SSM, they argued, was assessed as being too low and the fact that only a few missiles could be accommodated on a small ship did not make for good military sense. Nobody knew for certain and an operational test was probably required to settle the dispute.

1.2 THE SINKING OF INS ELATH [21 OCTOBER 1967]

The former Egyptian Minister of Information and renowned Arab journalist, Mohammed Heikal[3] holds the view that "after the Six Day War Israel made the great mistake of emphasising the conflict as one between itself and Egypt only. The way in which they treated the Egyptian wounded and prisoners; the way in which they projected their victory to the world; the way in which they subsequently behaved as they had a right to land their forces anywhere in Egypt they liked - all these were, from an Egyptian point of view, calculated to impose the maximum humiliation on their defeated enemy". He goes on to say : "They rubbed our noses in our defeat and showed over and over again what intense pleasure it gave them to do this".

The Israeli view[4], on the other hand, was that Israel was ringed by a vast Arab army. The Arab media promised the Israeli population destruction and annihilation in the

[3]M. Heikal, *The Road to Ramadan*, London : William Collins and Sons, 1975, pp. 38-48
[4]C. Herzog, *The War of Atonement*, Jerusalem : Steimatzky's Agency, 1975, pp. 1-7

most brutal manner. The horrors of the Nazi holocaust rose to the fore in the Jewish conscience. The Six Day War fired the psychosis further. After the Six Day War the Israelis regarded the canal as a defensible barrier between themselves and the Egyptians. A few weeks after the conclusion of the war, the first incidents had broken out along the Suez Canal front when the Egyptian forces began to harass the Israeli forces deployed along the canal. Fighting broke out at Ras el-Aish at the northern end of the sector between Port Said and Kantara. President Nasser's War of attrition had started. The scene was set for the first missile action in naval history.

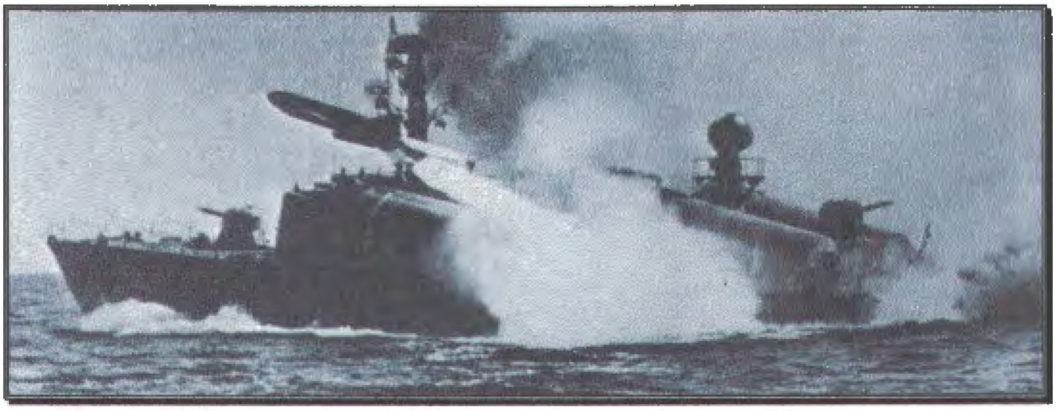


Figure 1.3 : SSN-2 being launched from an OSA I.

On[5] 21 October 1967 INS ELATH was patrolling about five nm off Port Said. Given the Egyptian sentiments, it is understandable that four SSN-2 missiles were fired at her from some KOMAR-class FAC(M)s, presumably alongside in Port Said. Three missiles hit, sinking the INS ELATH right away with drastic loss of life. The fourth missile had no target left to home on. The significance of this relatively unimportant historical fact is that the first SSMs was fired in anger ... and they hit their target. The operational test was successful.

1.3 THE ISRAELI COUNTER MEASURE [6][1967 - 1973]

The Israeli Navy, after its uninspiring record in the Six Day War, had sustained two tragic losses: the loss of the INS ELATH and the INS DAKAR, a submarine, that was lost with all hands en route from Great Britain to Israel in the east Mediterranean. The construction of a completely new navy was accordingly undertaken. When all twelve of the so-called Cherbourg FAC(M)s had arrived in Israel by December 1969, they were equipped with Israeli built Gabriel Mk 1 SSMs.

The Gabriel Mk 1 SSM was a fire-and-control missile with a range of 12 nm. It was developed by Israel in reply to the 5" guns mounted on the Soviet-built SKORY-class destroyers of the Egyptian Navy. These guns had a maximum effective range (MER) of 16000 yards. However, it was also necessary to develop tactics against the SSN-2 SSM with its superior range. The Israeli solution to this problem was reached through intensive development in three fields.

---

[5]C.W. Koburger, *Narrow Seas, Small Navies and Fat Merchantmen*, New York : Praeger, 1990, pp 63-65

[6]Herzog, *op.cit.* pp. 261-263

- Firstly, they developed ESM and ECM equipment to be used mainly to detect the enemy radar and missiles and then to interfere with the missiles.

- Secondly, they enhanced the manoeuvrability of their vessels.

- Thirdly, they enabled their vessels to strike home with missiles whilst approaching the enemy.

In addition the Israeli Navy decided to construct ships in Israel itself and adapt them to the area in which they were due to operate. Both the Mediterranean and the Red Sea required a longer range capability than that available in the Cherbourg FAC(M)s. By April 1973 the INS RESHEF, the first SAAR IV-class was operational, followed shortly thereafter by the INS KESHET[7]. These new ships, 59.1 metres long, displaced 415 tons standard and were armed with two 76 mm Oto Melara guns and seven Gabriel Mk 1 SSM.

1.4 THE YOM KIPPUR WAR [1973]

October 6, 1973 marked the tenth day of the Islamic Ramadan. On that day, in the year 623, the Prophet Mohammed began preparations for the Battle of Badr, which led ten days later to his triumphant entry into Mecca and the start of the spreading of Islam[8]. The combined Arab assault on Israel was thus code named Operation Badr. In Israel, the day blessed Yom Kippur, the holiest of the Jewish holidays. What a day to start a war!

Herzog's[9] account of the initial naval action in this war makes for exciting reading. The Syrian order of battle included three OSAs and six KOMARs. At the outbreak of the war, Israel was concerned that the Syrian FAC(M)s posed a greater tactical threat than the Egyptian ones. For that end they ordered a force of five FAC(M) including the INS RESHEF to deal with the Syrian Navy. The force sailed on the night of 6/7 October 1973 and patrolled the Syrian coast at a distance of 200 nm.

The next morning the force closed the Syrian coast. They were on an easterly course, heading for the town of Latakia when a Syrian minesweeper was sighted and INS RESHEF engaged it with a SSM. It sank almost immediately. But lying in wait to the south were three OSA FAC(M). The Israelis turned south to join the battle with the OSAs. On their approach, the Syrians fired a volley of missiles. Due to the Israeli counter missile tactics, none of the SSN-2 missiles hit.

The Israeli force sailed in paralleled columns and so manoeuvred that the Syrians found themselves sandwiched between the two columns. At 11:35 the Israelis joined in the battle. Both sides fired volleys of missiles. Within twenty five minutes the three

---

[7] *Jane's Fighting Ships 1974/75*, edited by Capt J.E. Moore. London : Jane's Yearbooks, [1974], p.181
[8] Insight Team of the Sunday Times, *Insight on the Middle East War*, London : André Deutch Ltd,1974, pp. 40-41
[9] Herzog, *op. cit.* pp. 263-266

Syrian ships were sunk. The Battle of Latakia, the first naval missile battle in history, had been won by the Israeli Navy without sustaining casualties. An analysis of the battle shows that the SSN-2 missiles were adequately disrupted by the Israeli electronic counter measures (ECM). The Syrians, with no ECM, did not stand a chance.

The main lessons learnt from the Battle of Latakia and the follow up operations by the Israeli Navy, were as follows :

- The Israeli ECM or the so-called soft kill option to deal with the SSN-2 missile was sufficient.
- The available gunnery did not aid the anti-missile tactics employed.

1.5 SUBSEQUENT DEVELOPMENTS [1974 - 1982]

The navies of the world took these lessons to heart and spent major parts of their respective budgets on developing soft kill tactics and the associated ECM equipment as well as developing AA weapons (or the so-called hard kill option) that could deal with the threat. Of the two, the latter soon became the more costly.

The place of the FAC(M) looked secure. The FAC(M) had a small radar cross section (RCS) and was thus easily protected by close range chaff. The ECM equipment that was developed by 1973 worked efficiently, for example, the range gate stealer (RGS) could entice the missile to lock onto it and break the missile lock on the target ship within approximately 20 seconds. At a missile speed of approximately Mach 0.9 or about 600 knots, if the missile was detected, the RGS started and the chaff deployed by the time the missile was still outside 3.6 nm, the RGS and chaff, coupled to some manoeuvring by the FAC(M) would, with a high probability, soft kill the missile.

$$Mach\ 1 \approx 38.925\sqrt{(\text{Air Temperature in } {}^0C + 273)}\ knots$$

$$\approx 0.33216\sqrt{(\text{Air Temperature in } {}^0C + 273)}\ m/s$$

Mach 0,9 is about 600 knots at sea level given an air temperature of 16°C

Figure 1.4: Rule of thumb for calculating the speed of sound at sea level[10]

The FAC(M) was also a highly affordable ship. By 1977 the price of a typical FAC(M) armed with six SSM was estimated in the region of $34 million. A frigate with the same armament as well as a torpedo and anti-submarine (TAS) capability were priced in the region of $125 million. The FAC(M) was the small navy's dream come true.

[10]Instruments, Magnetism and Compasses, *Senior Commercial Pilot Licence Study Notes*, Half Way House : Avex Air Training (Pty) Ltd, 1988 p 2.3.1

Navy	*SSM*
Canada	Canadian Sea Sparrow
France	Exocet MM 38 Exocet MM 40
International	ASSM Otomat
Israel	Gabriel
Italy	Sea Killer Mk 1 Sea Killer Mk 2 Mariner
Norway	Penguin
Sweden	RB 08A RBS 15
USSR	SSN-2 SSN-3 SSN-7 SSN-9
United States of America	Harpoon Tomahawk (Tactical)

Table 1.1 : Operational SSMs (1981)[11]

Much effort was expended to improve the SSM to deal with the soft kill option. The more modern missiles now boasted a leading edge tracker to negate the RGS and infra-red homing devices to address the chaff problem. The shipbuilders invested in spot jammers that would allow them to counter the leading edge tracker whilst they would endeavour to place as much chaff as possible in the vicinity of the ship and infra-red flares to be deployed with the chaff to counter the infra-red trackers.

Another development was the air launching of anti-ship missiles. In the main the missile producers modified their SSMs to be launched from aircraft. One such example is the French Exocet AM 39, a variant of the MM 38, that can be launched from, amongst others, the Super Etendard aircraft. The idea was to minimise the risk to SU-units and cash in on the FAC(M)s major weakness. It cannot defend itself against stand-off attacks by aircraft as it cannot engage the aircraft. The 76 mm gun carried on board some larger FAC(M)s is the maximum calibre to fit onto a ship of that size. Under well stabilised conditions the 76 mm gun, in the AA mode, normally has a MER of about four nm. Given the missile's range, the aircraft was able to fire the missile and return without putting itself at risk. (The French designation MM denotes *Mer-Mer* or Sea-Sea whilst the designation AM denotes *Air-Mer* or Air-Sea.)

For the purposes of this dissertation, these air launched variants will be regarded as SSMs. By 1981 the world's navies had produced eighteen different operational SSMs. These are listed in Table 1.1.

[11] *Jane's Weapon Systems 1980/81*, *op.cit.* pp. 794-795

1.6 THE FALKLANDS CONFLICT [1982]

Two aspects of the maritime operations that were conducted during the Falklands conflict of 1982 are noteworthy in this discussion. The first aspect is the sinking of HMS SHEFFIELD.

HMS SHEFFIELD[12] was one of the first Royal Navy's surface ships to reach the total exclusion zone promulgated earlier by the British government. Her role was to act as the forward air defence picket. However, she was ill equipped to deal any form of low level attack. Her main AA defence was the Sea dart missile. This missile is an excellent weapon against high flying targets, but was notoriously bad against low level targets. In fact, the Sea Dart cannot be used to engage targets under 2000 feet.

On Tuesday, 4 May 1982, SHEFFIELD was at defence stations, the second highest state of readiness with half the ship's company closed up at their stations. Just after lunch the operations officer, Lt Cdr. Nick Batho, picked up the briefest of radar contacts to the west. He had no possible indication that it was one of a pair of Argentinean Super Etandards fixing the SHEFFIELD's position. Only the radar information was duly relayed to the bridge.

Up on the bridge, the officer of the watch, Lt Peter Walpole, decided not to call the ship to action stations. This decision, although questionable from a readiness point of view, saved many lives as the alleyways would have been full of people en route to their action stations when the Exocet AM 39 hit the SHEFFIELD a few minutes later.

Walpole and the rest of his watch were scanning the horizon when they saw a puff of smoke in the distance. Nobody recognised it for what it was ... an oncoming missile. Some speculation would have it that at the time the ESM equipment was switched off to allow satellite communications with Whitehall. The story continues that satellite communications interfered with the working of the ESM equipment and it was standard practice in SHEFFIELD to switch the equipment off whilst using the satellite communications system.

The Exocet hit the ship at an oblique angle, penetrated the hull and exploded in the main engine compartment. The effect was devastating. A large part of the ship caught fire, some personnel were left dead or wounded and the electrical power on board was seriously disrupted. All fire fighting efforts failed and eventually the Officer Commanding, Captain (Now Rear Admiral) Sam Salt gave the order to abandon ship. The hulk remained afloat for another six days after which it was sunk by demolition charges.

The first lesson learnt from the Falklands conflict is that when the ESM/ECM combination of equipment is not functioning, there is no chance of surviving an oncoming SSM.

---

[12]Insight Team of the Sunday Times, *The Falklands War*, London : André Deutch, 1982, pp 163-167

Figure 1.5 : HMS SHEFFIELD after she was hit by an Exocet AM 39

The second[13] noteworthy set of facts pertains to some missile engagements on the 30 May of that year. Of the three Exocet AM 39s launched on 30 May by the Argentineans, two missed their targets and the third was destroyed in flight by HMS AVENGER's 4,5" gun. The Exocet inflicts greatest damage when it penetrates a ship's side cleanly at an angle of 90 degrees; its efficiency decreases when the angle of strike becomes oblique. Thus, ships which turned into or away from the attack stood a better chance of surviving than those which remained on course. In so doing they also shortened their radar profile and this was on one occasion, believed to influence the missile in its ultimate choice of target.

The second lesson learnt from the Falklands conflict is that manoeuvre is important in surviving SSM attack.

1.7 THE GULF WAR - NAVAL CAMPAIGN [1991]

From 18 January 1991 onwards, a well co-ordinated US Navy-Royal Navy offensive was waged against the Iraqi Navy which, though small, had a significant number of fast attack craft armed with a mix of combat-proven Soviet and Western SSMs.

US carrier aircraft attacked the naval base at Umm Qasr and mined the channel leading to it, thus forcing out the small craft while bottling-up large ships. Once in the waters off Bubiyan Island, the ships were exposed to direct attack by American and British aircraft, Royal Navy destroyers and frigates Lynx helicopters.

Eleven days after the outbreak of the war, Iraqi ships attempted to break out of Umm Qasr en masse, apparently hoping to seek asylum in Iranian ports, but were continuously and repeatedly attacked. Those who survived this attempt repeated the experience in mid-February, thereafter, in all intents and purposes, the Iraqi Navy ceased to exist. A chronological list of events are found in Table 1.2.

---

[13]B. Perret, *Weapons of the Falklands Conflict*, Dorset : Blandford Press, 1982, pp 125-126

Date	Description
22 Jan	1 T-43 minesweeper was disabled by A-6Es. 1 patrol boat was disabled.
23 Jan	1 Al Qaddissayah-class tanker was disabled by A-6Es. 1 Winchester-class hovercraft was sunk by A-6Es. 1 Zhuk patrol boat was sunk by A-6Es.
24 Jan	1 Zhuk patrol boat was sunk by A-6Es. 1 Spasilac salvage ship was sunk by A-6Es. 1 minelayer was sunk by a Harpoon from a Saudi ship. 1 minesweeper was sunk by a mine while evading an A-6E. 4 ships were struck at Umm Qasr naval base.
25 Jan	1 minelayer was hit while laying mines near sea oil terminal.
26 Jan	1 patrol boat was hit in Kuwait harbour. 1 TNC-45 craft was struck and left burning by A-6Es.
27 Jan	1 ship was sunk by A-6Es.
29 Jan	17 small boats were attacked; 4 sunk, 12 damaged. 1 large patrol boat was sunk by Sea Skua from HMS CARDIFF.
30 Jan	8 attack boats were struck, 4 were sunk, 3 were damaged, including Osa missile boats. Landing craft were also possibly hit and damaged. 1 T-43 hit by Sea Skua from HMS GLOUCESTER, left burning. 1 TNC-45 was hit by Sea Skua from HMS GLOUCESTER, left burning. 3 LSMs were sunk by Jaguars and A-6Es.
01 Feb	1 patrol boat was left burning at Min-al-Bakr oil terminal by A-6E
02 Feb	1 missile boat was hit by 2 laser-guided bombs and a second was possibly hit, at Al Kalia naval facility. 3 patrol boats were struck; 1 destroyed, 2 damaged. 1 patrol boat was destroyed by A-6Es in Kuwait harbour.
08 Feb	1 training ship and a TNC-45 patrol boat were struck by A-6Es at Cor-al-Zubayr.
09 Feb	1 Zhuk was damaged by a Rockeye from an A-6E near Faylakah Island.
10 Feb	2 patrol boats were sunk by A-6Es in the northern Gulf.
14 Feb	1 Osa missile boat was sunk in Kuwait Bay by A-6Es.
20 Feb	1 gunboat bombed by S-3 aircraft.

Table 1.2 : Destruction of the Iraqi Navy[14]

Considering the frail nature of the FAC(M) as opposed to a Destroyer or Frigate, it is obvious that Iraq's FAC(M)s could not survive a missile hit. Another aspect is the fact that none of the Iraqi ships had any ESM or ECM equipment worthy of note. Their soft kill ability could have been considered as being naught. Sadam Hussein's armada of FAC(M)s did not stand a chance.

1.8 THE FAC(M) DILEMMA TODAY

The demise of the Iraqi Navy suggested that the FAC(M) was an overrated ship that would fall prey to her bigger sisters in any operational scenario. On the other hand, there is a question mark over the Iraqi commanders' willingness to fight in the Gulf War. Other countries, such as South Africa, operate FAC(M)s that are more advanced in terms of weaponry and electronic equipment. They still believe in the concept and see these ships as akin to the destroyers in the turn of the century.

[14]B.W. Watson,... [et. al.], *Military Lessons from the Gulf War*, London : Greenhill Books, 1991, pp 261-262

In the South African scenario, her neighbours and for that matter most of the countries on the African western and eastern seaboard, cannot compete with the South African Navy (SAN). In the medium to long term, any potential foe will have to be from a country outside of Africa. The country is at present at peace with all other nations but one can sketch the following two scenarios :

1. India wishes to play the role as the superpower in the Indian Ocean and comes into conflict with South Africa.

2. Brazil wishes to enhance her influence in the South Atlantic and comes into conflict with South Africa.

The latter is most probably the more likely scenario as Brazil is presently engaged in training Namibian sailors and is making overtures to the South African Military from an assumed position of leadership. In both the scenarios a fleet consisting of so-called blue-water ships will confront the South African Navy.

Rear Admiral Johan Retief[15], at the time, Chief of Naval Operations of the South African Navy is on record as having the following views in regard to FAC(M)s :

- When FAC(M)s operate against frigates or ships of that size, the frigate has the detection advantage. Therefore, the frigate will in all likelihood fire its SSMs first.

- For a FAC(M) to be able to engage such ships successfully, it is a prerequisite that the FAC(M) must be able to counter the oncoming missiles.

- In the event that one of the above scenarios becomes reality, the South African FAC(M) will more likely encounter fire-and-forget SSMs than other types of missiles.

In a betting game as described by Clemen[16] (see Figure 1.7), the Admiral scored his subjective probability that the South African missile counter tactics will be effective as indicated in Table 1.3. Furthermore, for nine Exocet MM 38 fired against a pair of South African FAC(M)s the cumulative distribution function is in Figure 1.6.

---

[15]Retief, Rear Admiral J.F. Personal Interview. 12th February 1996, Pretoria.
[16]R.T. Clemen, *Making Hard Decisions : An Introduction to Decision Analysis*, Boston : PWS-Kent, c1991, pp 210-213

SSM	*P(SAN Tactic Successful)*
SSN-2B	99,8%
Exocet MM 38	63,1%
Harpoon	54,0%

Table 1.3 : Subjective probabilities of the SA Navy's missile counter measures being effective.

Given the historical background and the views expressed by the Chief of Naval Operations, the conclusion that may be reached is that it is necessary to ensure that the effectiveness of one's missile counter tactics is maximised.

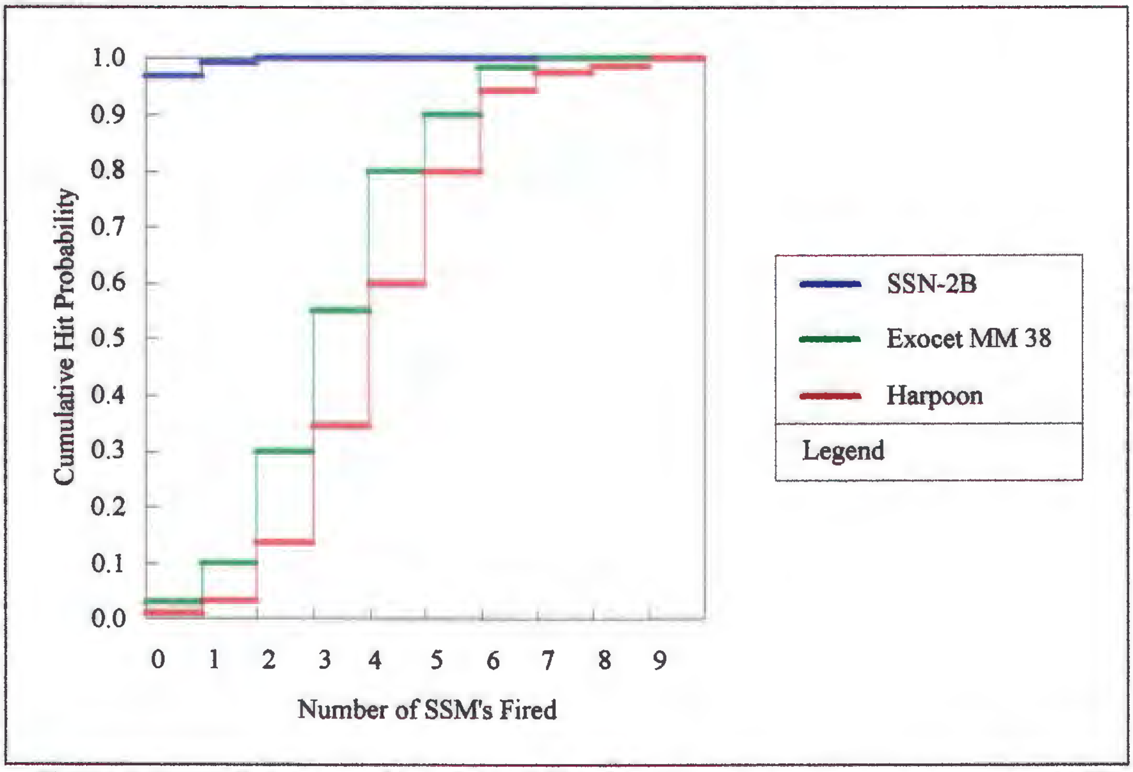


Figure 1.6 : Cumulative Distribution Function showing subjective dominance of Harpoon over Exocet MM 38 over SSN-2.

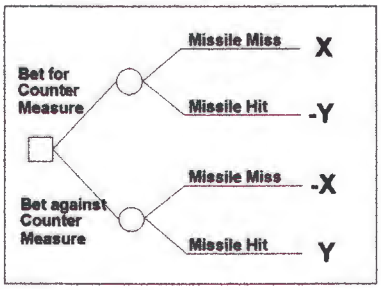
1.9 SOLVING THE PROBLEM

To maximise the effectiveness of a particular military tactic may best be described as a complex matter. If you want to maintain a leading edge in war, you will not publish your country's secrets, nor may any student claim that he has the right to do so. On the other hand, such problems are exciting academic challenges. Not to be allowed to tackle these, would stifle academic thought.

A method to assess a subjective probability is to ask about the bets that the decision maker would be willing to place. The idea is to find a specific amount to win or to lose such that the decision maker is simply indifferent as to which side of the bet to take. Once those amounts are established, the expected value of the bet must be the same, regardless of which side is taken. Given these conditions, we can the solve for the subjective probability. The Chief of Naval Operations had to bet on one of the following :

Bet	***Outcome***
Counter measure is successful	***Win R X if the missile misses*** ***Lose R Y if the missile hits***
Counter measure is unsuccessful	***Lose R X if the missile misses*** ***Win R Y if the missile hits***

A decision tree representation for assessing subjective probability via the betting method is shown below.



If the decision maker is indifferent as to the two bets, then, in his mind, the expected values of the two bets must be equal.

$$\therefore \quad X\,\text{P(Counter Works)} - Y[1-\text{P(Counter Works)}]$$
$$= -X\,\text{P(Counter Works)} + Y[1-\text{P(Counter Works)}]$$

which implies that:

$$X\,\text{P(Counter Works)} - Y + Y\,\text{P(Counter Works)} = 0$$

Collecting terms deliver:

$$(X + Y)\,\text{P(Counter Works)} - Y = 0$$

which reduces to:

$$\text{P(Counter Works)} = Y/(X+Y)$$

Figure 1.7 : Assessing subjective probability via the betting method.

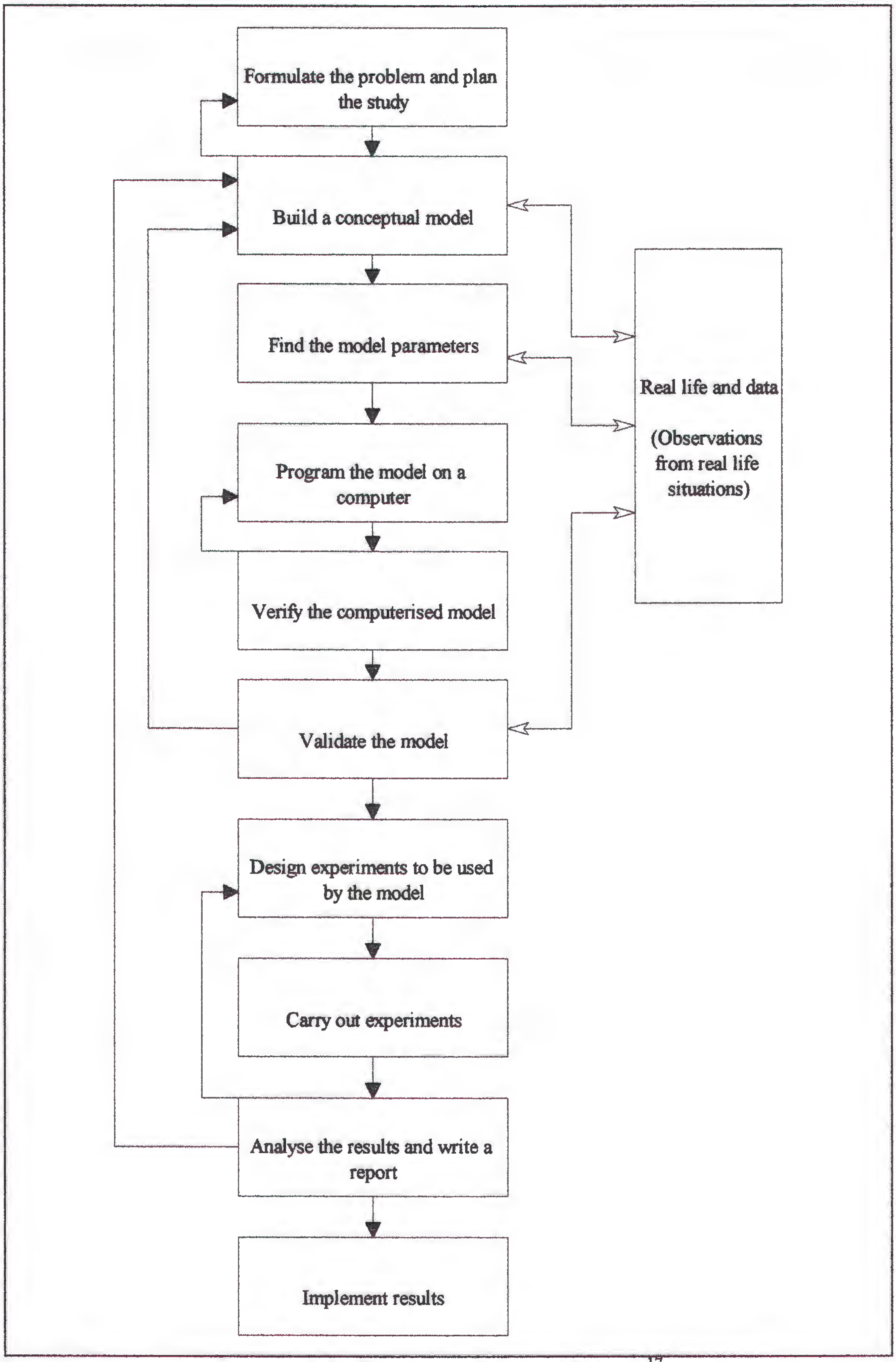


Figure 1.8 : Simulation Process Diagram[17]

[17]G.N. Engelbrecht, *Numeric Simulation Techniques*, Simon's Town: SA Navy. 1996. P 7.

The problem of maximising FAC(M) counter measures or tactics against such missiles as the Exocet family, Harpoon and Otomat is also a very challenging one. The ship, missile and the influence of the environment on them are complex issues in themselves. Putting them together in one model is a worthy but difficult exercise.

In order to heed security considerations and to be able to tackle this challenging academic task, the following work method was devised :

> *The aim of this dissertation is to show how counter missile tactics can be optimised. As a result the ship and missile will be modelled as generic concepts while the environment will be a chosen area of operations. The applicable methodology is to simulate the ship, missile and environment as well as the interactions between them. At the same time, the ship will be carrying out several missile counter measures. The methodology will then be to build a dynamic simulation model to optimise soft kill tactics by a generic FAC(M) against a generic surface-to-surface SSM in the chosen environment.*

The simulation will be developed against the framework of the simulation process diagram in Figure 1.8. The lay-out of the dissertation will follow the process and the various steps will be dealt with in that sequence. However, more than one chapter may be necessary to develop a particular step in the process. The concept model is central to the whole issue and chapters two through to five will be set aside for that.

Where real life data is of a classified nature, such data will be approximated by information of a theoretical nature. An example is the tracking algorithm of a missile. This body of knowledge would be classified "Top Secret" by any Navy, but references to the techniques that might be employed can be found in open sources. A "typical" algorithm will then be developed for the purposes of the dissertation.

Chapter 2

CONCEPT MODEL OF THE ENVIRONMENT

2.1 AREA OF OPERATIONS

In deciding which area of operations is most applicable, we must consider the following two factors:

- Within the context of this dissertation, the area of operations should be applicable to the South African Naval situation.

- We assume that SSMs are normally designed to be used in any sea conditions for at least 95% of the time. A conservative lower limit for this assumption would be to consider an area which, in turn, can be considered to be the worst case scenario in regard to prevailing swell, wind waves and wind. In this worst case scenario, the missile operating envelope must be such that it can fly over the sea, detect its target and successfully persecute same for at least 95% of the time.

2.1.1 THE SOUTH AFRICAN MILITARY SITUATION

The South African Navy forms an integral part of the South African National Defence Force (SANDF). As such, it is bound by the civil control of the South African Parliament. Parliament's policy is that[1] South Africa will maintain a defensive military posture. In turn, the SANDF will thus act as a defender of South African sovereignty. However, this does not imply that the SANDF may not act offensively. In order to defend the Republic, the SANDF may very well decide, with parliament's sanction and in line with von Clausewitz's dictum[2] that "A swift and vigorous transition to attack - the flashing sword of vengeance - is the most brilliant point of the defensive", to act offensively.

2.1.2 NAVAL AREA OF OPERATIONS

The SAN will follow the strategy and operations policy of the SANDF. As such, the primary area of operations for the Navy could well be the eastern and western African seaboard south of the equator. This area of operations will extend at least to the half-operating radius of the Navy's ships. This implies the area bounded by the equator,

---

[1] Republic of South Africa, Parliament, *White Paper on Defence*, 1996, p6.
[2] R.D. Heinl, *Dictionary of Military and Naval Quotations*, Annapolis: Naval Institute Press, 1966. p83.

the Greenwich meridian, 45°S and 50°E. For the purpose of this dissertation, we choose this area as the chosen area of operations.

2.2 THE WEATHER IN THE CHOSEN AREA OF OPERATIONS

We have assumed in Section 2.1 that SSMs are normally designed to be used in any prevailing sea conditions in our chosen area of operations for at least 95% of the time. In order to model the sea in our chosen area of operations and bearing in mind the fact that average sea conditions as well as their variances differ from place to place, we will find a small area where the worst case scenario in regard to swell and wind waves exists and find the 95th percentile to determine the missile operating envelope and the typical wind conditions for inclusion in our simulation.

2.2.1 SWELL HEIGHT

The average swell heights in the chosen area of operations are depicted in Figure 2.1[3]. The averages are shown for smaller blocks measuring one degree of latitude and one degree of longitude at a time.

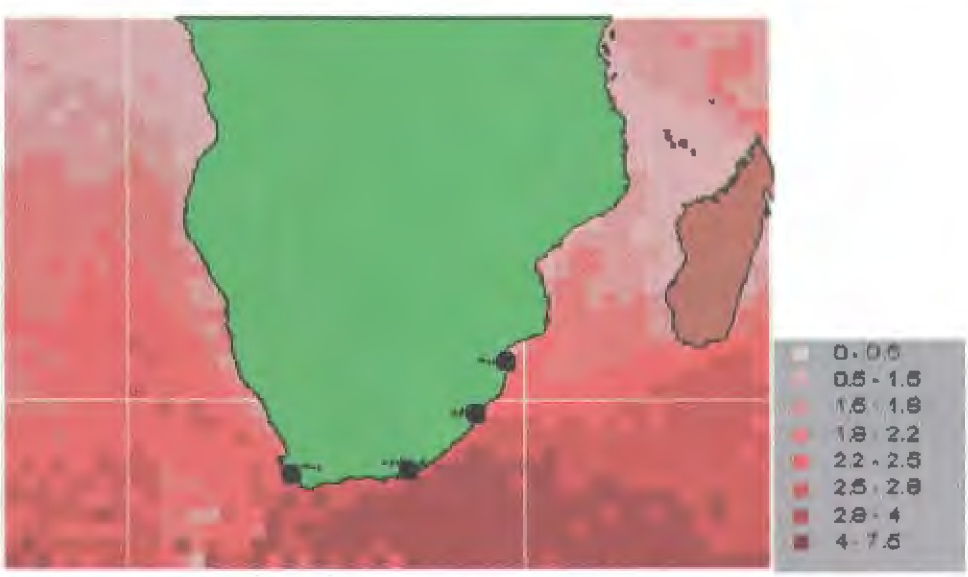


Figure 2.1: Average swell conditions on the Southern African seaboard in metres.

By inspection we find the worst case scenario to be an average swell height of between 4 and 7.5 metres. The area to the south - east (SE) of Port Elizabeth is by-and-large such an area.

The variance for the swell height for the corresponding areas is depicted in Figure 2.2. Again, by inspection, we find the worst case scenario, that is, a variance in swell height of between 2 and 5 metres, SE of Port Elizabeth. However, this time the area

[3] All data in this section from the South African Maritime Data Centre for Oceanology (SAMDCO) Database administered for the SA Navy and others by the CSIR.

concerned is much smaller and lies roughly within 120 nm to seaward of Port Elizabeth and its adjoining land mass.

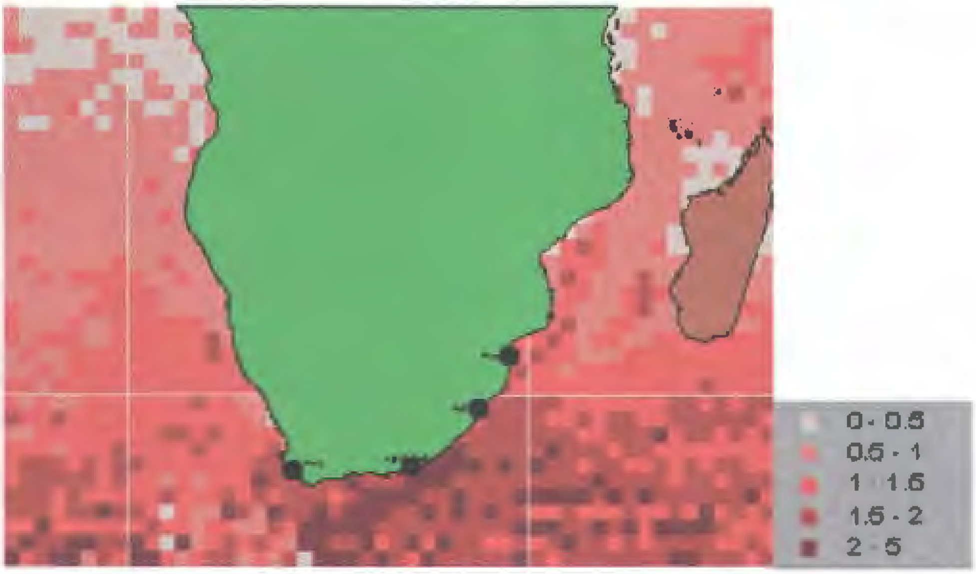


Figure 2.2: Swell height variance on the Southern African seaboard in metres.

2.2.2 WIND WAVES

Whilst swells are the result of distant influences on sea conditions, wind waves are the result of local weather on sea conditions. Average wind waves for the chosen area of operations are found in Figure 2.3 whilst the applicable variance chart is in Figure 2.4.

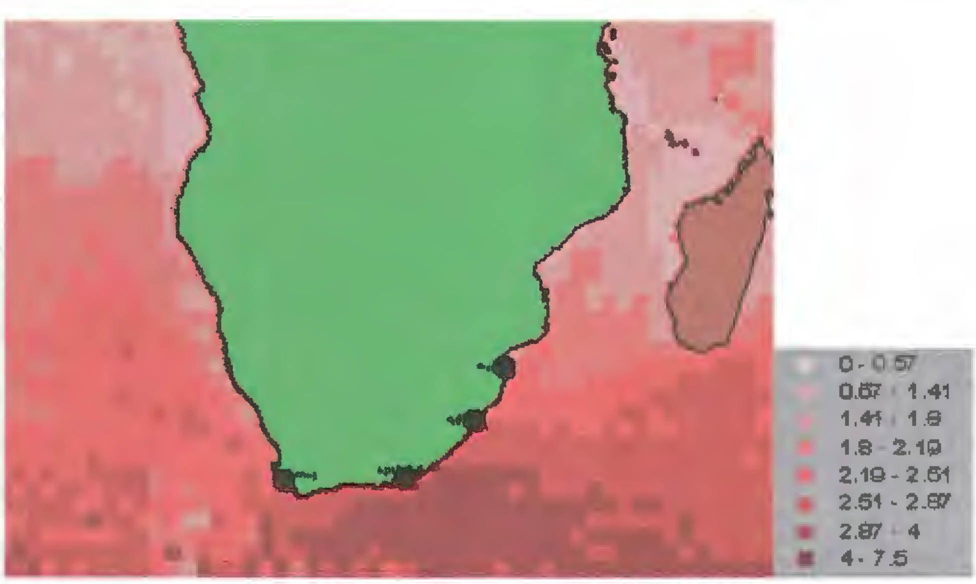


Figure 2.3: Average wind wave conditions on the Southern African seaboard in metres.

The pattern that emerges from Figure 2.3 and Figure 2.4 again points to the area SE from Port Elizabeth as the worst case scenario in the chosen area of operations.

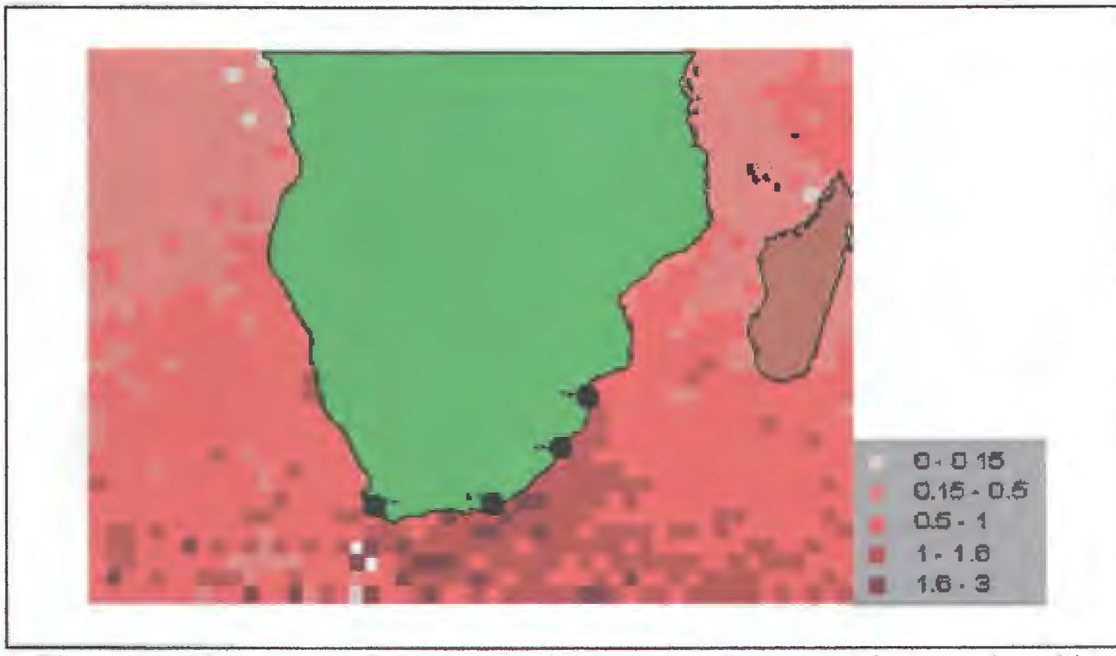


Figure 2.4: Variance of wind wave conditions on the Southern African seaboard in metres.

2.3 THE SIMULATION AREA OF OPERATIONS

We define the worst case scenario, that is, the area to the SE of Port Elizabeth, as the area bound by latitudes 34°S and 35°S and the longitudes 24°E and 26°E. Henceforth we will call this area the *simulation area*.

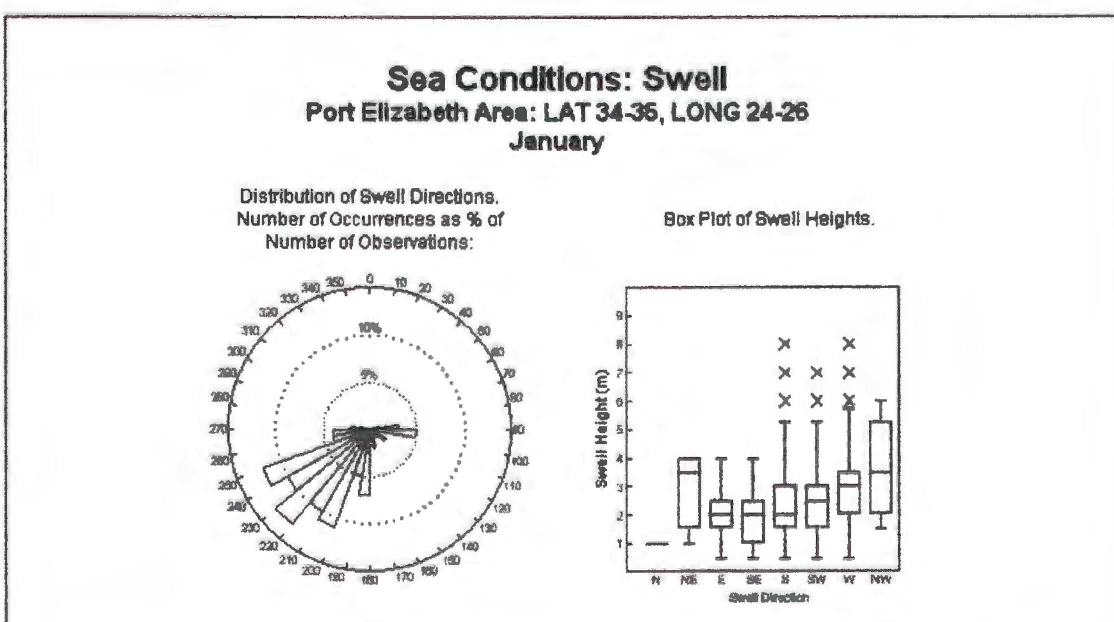


Figure 2.5: Swell Conditions off Port Elizabeth for January.

Weather patterns and the resultant sea conditions also experience seasonal changes. The weather in our worst case scenario area is worst in summer. From the Maritime Data Display[4] we find the worst summer month for the area. The swell situation in the area is depicted in Figure 2.5 whilst the wind wave situation is depicted in Figure 2.6.

Now, consider Figure 2.5. By inspection we note that the box plots for swell heights in regard to different swell directions indicates that the distribution of these are not symmetric around the mean. Furthermore, for various directions of swell, the distributions of swell heights are different for various swell directions.

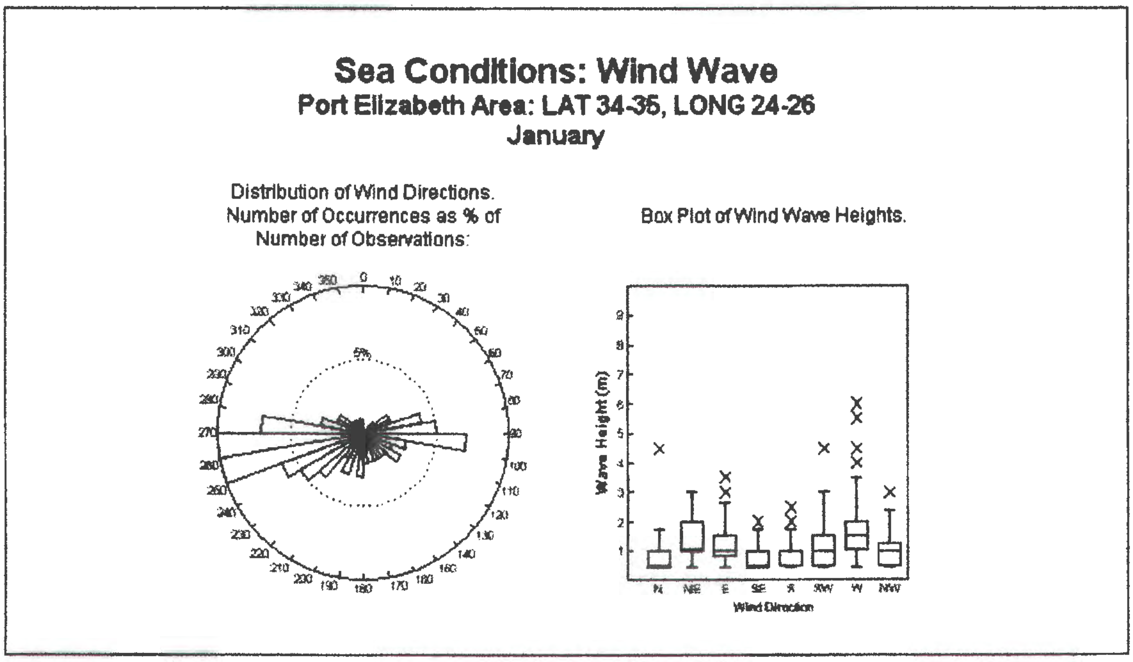


Figure 2.6: Wind wave conditions off Port Elizabeth for January.

By inspection of Figure 2.6, we note that a similar situation exists for wind waves. Furthermore, the SAMDCO database exists after several thousands of observations have been taken over the last twenty years. These observations were taken by seafarers who plied South African waters. In general, these sea conditions were estimated and reported by these seafarers to the relevant authorities ashore, that is, these observations are estimates and not measured data.

For this situation, it is prudent to use some rank statistics to describe the required information as the data is not so distributed that it can be otherwise analysed. For that end, the SAMDCO database was revisited for the simulation area and the following height for swell and wind wave were found at the 95th percentile for 21 798 and 25 378 records respectively:

- Swell Height: 5.0 metres.
- Wind Wave Height: 3.5 metres.

[4] By the kind permission of the SAN Hydrographic Office.

From this can argue that in the simulation area we will find that 95% of the time the swell height would be less than or equal to 5.0 metres and the wind wave height will be less than or equal to 3.5 metres. However, the sea state is a combination of swell and wind wave. If we add the two statistics, we can say that 8.5 metres is an adequate, but somewhat stricter, estimate of the sea state in the area under consideration for at least 95% of the time. This estimate can be regarded, in a statistical sense, as even more strict if it were applied to the whole chosen area of operations as we have only considered the worst case scenario. Thus we choose 8.5 metres as an adequate estimate for the sea state at the 95th percentile in the simulation area.

2.4 THE SIMULATION ARENA

We have defined the simulation area as the area bound by latitude 34°S to latitude 35°S and longitude 24°E to longitude 26°E. Now, within the simulation area is a smaller area which we will call the *simulation arena* or just *arena* where the actual simulation will take place. This area is akin to a battlefield arena, that is, the area within the area of operations where the actual battle takes place.

The simulation arena includes the ship and the missile's initial position (IP) as well as the area where they will subsequently move in. We will choose the arena in a simple random manner. Although, this choice is not apparent, it is implied by the manner in which factors such as current and wind is chosen. Furthermore, we describe the arena as a Cartesian co-ordinate system or grid to facilitate the calculation of the positions of the various objects in the simulation and the resolution of the missile range gate. Also, we will follow the convention that the ship's IP at the commencement of every simulation replication will be the position (0,0) in the grid. (Also see Section 3.4.1)

Now, we define a *polar vector* as a vector described by a bearing, course or direction in true compass degrees (°T) and a distance or velocity in nm or knots respectively. Furthermore, we define the simulation arena as a Cartesian grid in metres and we define vectors which consist of a horizontal (east) and a vertical (north) component as a *Cartesian vector*, for example, the Cartesian vector (1,1) indicates a vector of magnitude one in an easterly direction and magnitude one in a northerly direction. This vector is equivalent to a polar vector with direction 045°T and magnitude of $\sqrt{2}$. In order to transform vectors from polar to Cartesian, the following constants shall be used:

- 1 knot = 0.512 m/s.
- 1 nm = 1843.2 m.

2.5 WIND MODEL

We again select the simulation area and as a result, the month of January, to develop a wind model for our simulation. The Maritime Data Display summarises wind conditions for the simulation area and the month of January as depicted in Figure 2.7.

As these wind conditions are concurrent to the wind wave conditions, this summary is accepted as the basis for the wind model. Note that whereas the wind direction data is given as a percentage of wind experienced in a spoke of ten degrees, the wind speed box plots are given for spokes of 45°, that is, 22½° either side of the primary and intermediate compass directions.

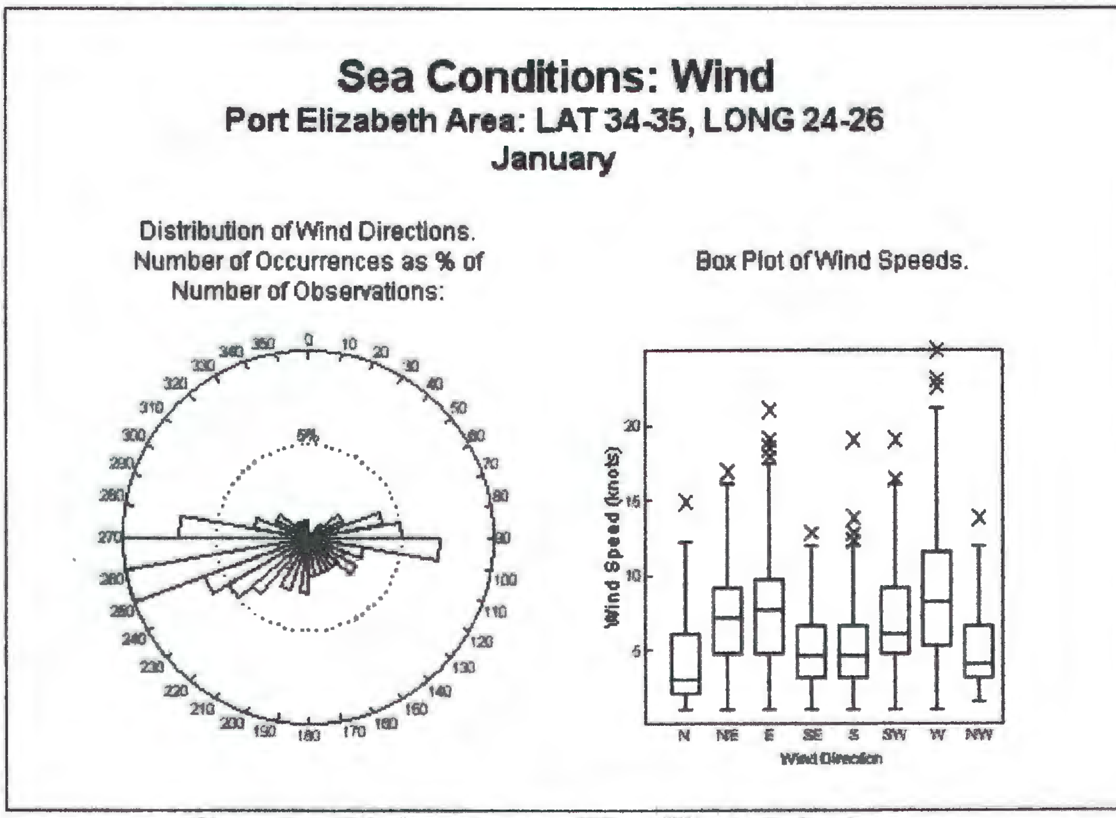


Figure 2.7: Wind conditions off Port Elizabeth for January.

In order to simplify the model, the outlying data items above the box plots are discarded. Maloney[5] shows that there is a correlation between wind waves and wind speed experienced and if we are only to consider sea states below the 95th percentile, then without compromising the validity of our model, we may discard the outliers.

2.5.1 WIND DIRECTION DISTRIBUTION

Note that the wind direction data described above is such that our data is grouped, that is, we have $k = 36$ adjacent intervals,

$$[a_0, a_1), [a_1, a_2), \ldots, [a_{k-1}, a_k],$$

and the jth interval contains n_j observations with $n_1 + n_2 + \ldots + n_k = n$. A piece-wise linear empirical distribution G was found by first letting $G(a_0) = 0$ and

$$G(a_j) = \frac{1}{n}\sum_{i=1}^{j} n_i \qquad (2.1)$$

[5] E.S. Maloney, *Dutton's Navigation and Piloting*, 13 ed., Annapolis: Naval Institute Press. 1978. p860.

for $j = 1, 2, \ldots, k$. The results are summarised in Table 2.1 and Figure 2.8.

Direction °T	$F(x)$	Direction °T	$F(x)$	Direction °T	$F(x)$
000	0.00				
010	0.03	130	0.57	250	0.66
020	0.04	140	0.58	260	0.70
030	0.07	150	0.59	270	0.75
040	0.08	160	0.60	280	0.82
050	0.12	170	0.61	290	0.85
060	0.17	180	0.61	300	0.87
070	0.23	190	0.61	310	0.90
080	0.33	200	0.61	320	0.92
090	0.43	210	0.61	330	0.94
100	0.50	220	0.61	340	0.96
110	0.53	230	0.62	350	0.98
120	0.55	240	0.64	360	1.00

Table 2.1: Cumulative distribution function for wind direction.

The exact empirical distribution function can be defined by interpolating linearly between the a_js which will result in

$$G(x) = \begin{cases} 0 & \text{if} \quad x < a_0 \\ G(a_{j-1}) + \dfrac{x - a_{j-1}}{a_j - a_{j-1}}\left(G(a_j) - G(a_{j-1})\right) & \text{if} \quad a_{j-1} \le x < a_j \\ 1 & \text{if} \quad a_k \le x \end{cases} \tag{2.2}$$

for $j = 1, 2, \ldots, k$.

The wind data in the SAMDCO database reflects the wind from where the wind blows. However, our model will require the actual vector so as to calculate the influence of the wind on objects such as the missile and chaff clouds.

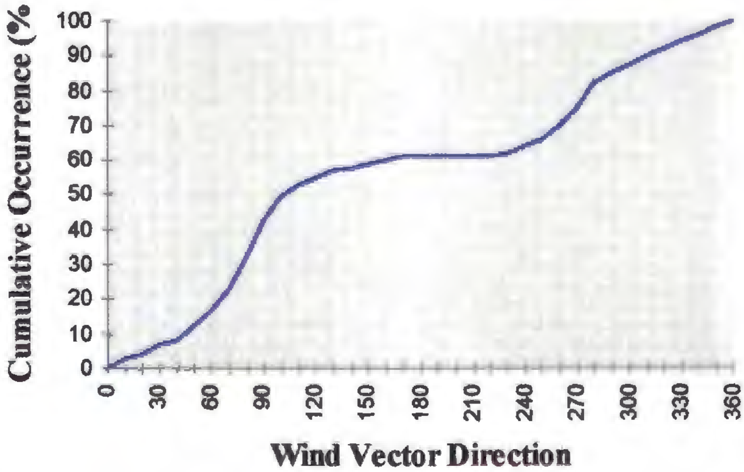


Figure 2.8: Cumulative Distribution Function for Wind Direction.

In order to generate random numbers from G, we use the inverse-transform method described by Kelton and Law.[6] The method is described in Algorithm 2.1.

Step 1:	Generate $U \sim U(0,1)$.
Step 2:	Find the non-negative integer J where $0 \le J \le k-1$ and such that $G(a_J) \le U < G(a_{J+1})$.
Step 3:	Return $X = a_J + \frac{a_{J+1} - a_J}{G(a_{J+1}) - G(a_J)}\left(U - G(a_J)\right)$.

Algorithm 2.1: Generation of random numbers to represent wind direction.

Note that there are no wind vectors between 170°T and 220°T. Algorithm 2.1 will ensure that no such vectors are generated because the J found in step 2 satisfies the inequality $G(a_J) < G(a_{J+1})$ so that no X can be generated in an interval for which there were no observations.

2.5.2 WIND SPEED DISTRIBUTIONS

The wind speed distributions for the directional wind polar vector parts are derived from the box plots in Figure 2.7 and are tabulated in Table 2.2. Each distribution holds for 22½° on either side of the associated directional wind polar vector part.

Direction °T	$F(x) =$ 0	$F(x) =$ 0.25	$F(x) =$ 0.5	$F(x) =$ 0.75	$F(x) =$ 1
000	0.0	3.0	4.5	6.5	12.0
045	0.0	5.0	6.0	9.0	16.0
090	0.0	5.5	8.0	11.5	21.0
135	0.5	3.0	4.0	6.5	12.0
180	0.0	2.0	3.0	6.0	12.3
225	0.0	4.8	7.0	9.0	16.0
270	0.0	4.8	7.5	10.0	17.0
315	0.0	3.0	4.5	6.5	12.0

Table 2.2: Empirical wind speed distributions in knots.

We note that, for a given wind direction, the distributions in Table 2.2 are such that the values of $F(x)$ increase by 0.25 at every subsequent point on the graph of $F(x)$ where $F(x)$ might change its slope. This observation allows us to use an inverse-

[6] A.M. Law and W.D. Kelton, *Simulation Modeling and Analysis(sic.)*. 2 ed. New York: McGraw-Hill, 1991. p495.

transform method for generating the applicable wind speeds without using the search method described in Algorithm 2.1.[7] The method is given in Algorithm 2.2.

Step 1:	Generate $U \sim U(0,1)$.
Step 2:	Set $P = (n-1)U$. Set $J = \lfloor P \rfloor + 1$.
Step 3:	Return $X = a_J + (P - J + 1)(a_{J+1} - a_J)$.

Note that for our model we have that $n = 5$ and the a_j are the n inflection points (Afr: knakpunte) of the distribution.

Algorithm 2.2: Generation of random numbers to represent wind speed[8].

Given an output from Algorithm 2.3, in order to decide which set of data in Table 2.2 is applicable, the decision criteria in Table 2.3 must be used.

Decision Interval	337½ -22½	22½ - 67½	67½ - 112½	112½ -157½	157½ -202½	202½ -247½	247½ -292½	292½- 337½
Choose row for	000	045	090	135	180	225	270	315

Table 2.3: Decision criteria to choose a particular distribution from Table 2.2.

2.5.3 WIND SIMULATION MODEL

In order to handle position information in the simulation, wind velocity will be expressed in metres per second (m/s). However, in order to handle direction information in the computer to calculate new positions in the grid, we shall use radians as a measure of angle. Where our input information is in degrees, we must make the transformation to radians at an appropriate place in the program.

Furthermore, in order to reduce the variance between simulations to evaluate different tactics, common random numbers will be used. At this stage, it will suffice to say that, for that end, every user of generated uniformly distributed random numbers over the interval [0,1) will be allocated a separate row of such numbers. The computer variables `WindDirectionStream` and `WindSpeedStream` indicate such allocated rows of random numbers.

The wind distribution data, in the appropriate format, is kept in a file, WIND.DAT, and is read during the initialisation of the global variables in the simulation. It is contained in one record with the following self-explanatory data structure:

```
type  windtype = record
                    direction    : array [1..36] of real;
                    speed        : array [1..8,1..5] of real;
                 end;
```

[7] Law and Kelton. *op. cit.* pp494-495.
[8] G.N. Engelbrecht, *A Guide to Numeric Simulation Techniques*. Simon's Town: SA Navy, 1996. p99.

The wind data is kept in the global variable `wind`. The method to generate the wind for a particular replicate of the simulation is as follows:

- Generate the wind direction as described in Algorithm 2.1.
- Generate the wind speed as described in Algorithm 2.2.
- Transform the wind direction and speed into a Cartesian vector.

Procedures and functions, in Turbo Pascal 6.0, used to carry out the method described above are in Appendix G at the pages as indicated. They entail the following functions and procedures:

Function/Procedure	Name	Pages
`function`	`radian`	G-7
`function`	`CartForm`	G-8
`procedure`	`CalculateVector`	G-23
`procedure`	`GetWindDirection`	G-24
`procedure`	`GetWindSpeed`	G-24
`procedure`	`GenerateWindVector`	G-25

2.5.4 VALIDATION OF THE WIND MODEL

The output of the wind model was tested separately at the 5% level for 3 000 replications for wind direction and wind speed and no significant difference between the output for the two wind parameters and the expected values from the assumed empirical distributions could be found.

Note that all the $U \sim U(0,1)$ used in the simulation will be generated by a Pascal version of a prime modulus multiplicative linear congruential generator based on the Marse and Roberts generator[9] instead of by using the in-built compiler function. This random number generator has been tested thoroughly[10] and is regarded as superior to most in-built compiler functions which include the one in the Turbo Pascal version 6.0 compiler.

2.6 CURRENT MODEL

The prevailing current experienced influences the ship's track made good over the ground in a similar manner as the wind influences the flight path of the missile. However, very little data on the prevailing currents in the simulation area is available.

[9] Engelbrecht, *op. cit.* pp30-32.
[10] Law and Kelton, *op. cit.* pp449-456.

The Africa Pilot[11] give some insight into the problem. According to Figure 2.9, the current in the simulation area runs in a south-westerly direction at about 1½ knots.

2.6.1 CURRENT VELOCITY

However, a further study of the Africa Pilot[12] reveals that although tidal currents are generally weak in the simulation area, the Agulhas current velocity can be as strong as five knots, especially after some NE winds on the Natal coast. To the west of our area the Agulhas current splits into northern and southern sub-currents. The velocity of these two sub-currents is about half of what is encountered to the east of the simulation area. Now, as a result of the aforesaid, we assume a triangular current velocity distribution with the least current velocity, ¾ knots, the most likely current velocity to be 1½ knots and the uppermost current velocity to be five knots.

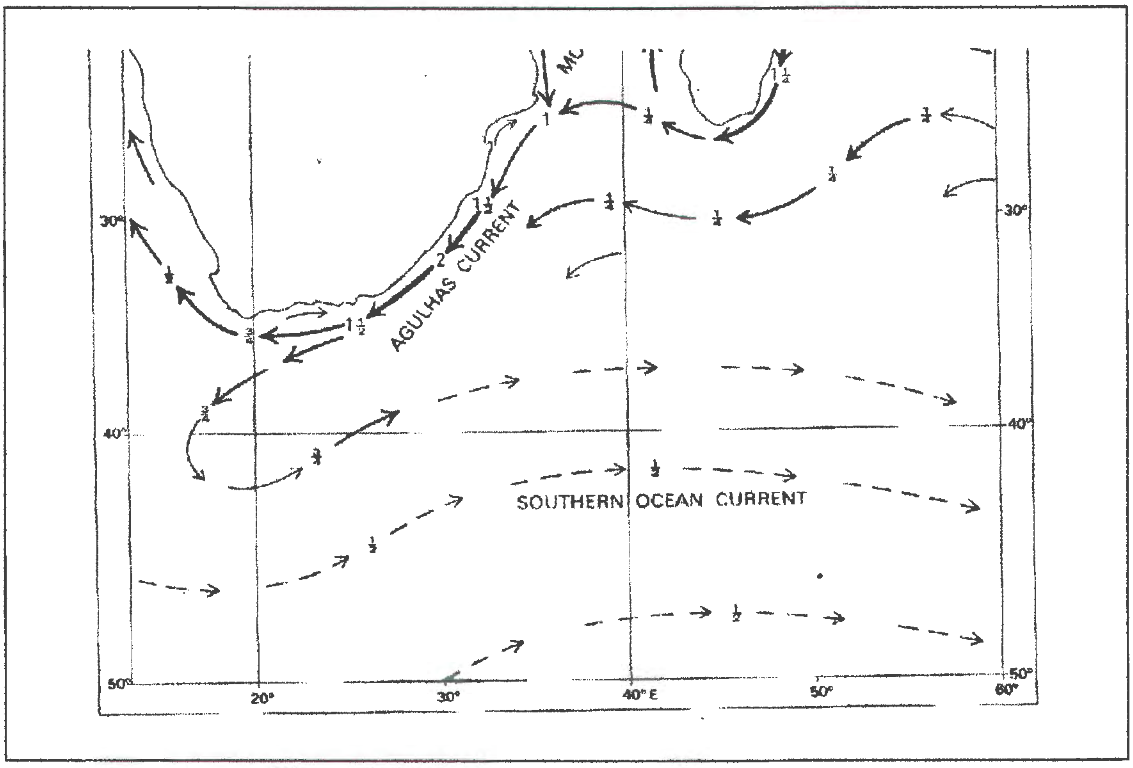


Figure 2.9: Vector-mean currents for January.

Now, if we denote the least current velocity a, the uppermost current velocity b and the most likely current velocity c then the triangular distribution for current velocity, $F_{CV}(x)$ is expressed as follows[13]:

---

[11] *Africa Pilot Volume III, South and East Coasts of Africa from Cape Agulhus to Ras Binnah, including the Islands of Zanzibar and Pemba*. 13 ed., Taunton: Hydrographer of the Navy, 1980. p27.

[12] *Ibid.* p30.

[13] Engelbrecht, *op. cit.* p60.

$$F_{CV}(x) = \begin{cases} 0 & \text{if} \quad x < a \\ \dfrac{(x-a)^2}{(b-a)(c-a)} & \text{if} \quad a \le x \le c \\ 1 - \dfrac{(b-x)^2}{(b-a)(b-c)} & \text{if} \quad c < x \le b \\ 1 & \text{if} \quad x > b. \end{cases} \tag{2.3}$$

Furthermore, the expected mean value of the current velocity is $(a + b + c)/3$, that is, about 2.4167 knots. Again, note that the simulation will handle current velocity in m/s.

2.6.2 CURRENT DIRECTION

By inspection of Figure 2.9, it was noted that the current in the simulation area flows in a south-westerly direction with the northern most limit being about 250°T and the southern most limit being about 210°T. The distribution of current direction is not known and, as there is little, or no data available to estimate such a distribution, it is assumed that current direction is uniformly distributed between the limits set above. The southern most current direction limit is denoted a and the northern most current direction limit is denoted b. Then the distribution function for current direction, $F_{CD}(x)$, is expressed as follows[14]:

$$F_{CD}(x) = \begin{cases} 0 & \text{if} \quad x < a \\ \dfrac{x-a}{b-a} & \text{if} \quad a \le x \le b \\ 1 & \text{if} \quad b < x. \end{cases} \tag{2.4}$$

2.6.3 CURRENT MODEL

Random numbers, to represent both current direction and current velocity, can be generated by the inverse-transform method[15]. In order to use the inverse-transform method for generating triangular distributed variates, the interval $[a,b]$ must first be scaled to fit the interval [0,1]. Thereafter the inverse-transform method can readily be used to generate a triangular distributed number from the interval [0,1]. On completion, the obtained random number must be scaled to fit the interval $[a,b]$. The Turbo Pascal 6.0 procedures and functions for the current model is given directly.

```
function uniform(a,b : real; stream : integer) : real;
{--------------------------------------------}
{ Returns a uniform distributed random number }
{--------------------------------------------}
   begin
      uniform := a + (b-a)*rand(stream);
   end;
```

[14] Engelbrecht, *op. cit.* p55.
[15] Law and Kelton. *op. cit.* pp485-486, 494.

```
function triang(a,b,c : real; stream : integer) : real;
{----------------------------------------------}
{ Returns a Triangular-distributed random number }
{----------------------------------------------}
var U,X,c1 : real;

begin
   U  := rand(stream);
   c1 := (c-a)/(b-a);
   if NOT(U>c1) then
      begin
         X := sqrt(U*c1);
      end
   else
      begin
         X := 1 - sqrt((1-c1)*(1-U));
      end;
   triang := a + (b-a)*X;
end;

procedure GenerateCurrent(var current : vectortype);
{------------------------------------------------}
{ Procedure to model current as a Cartesian vector }
{------------------------------------------------}
var v : vectortype;

begin
   {-------------------}
   { Current Direction }
   {-------------------}
   v[1] := uniform(210,250,CurrentDirectionStream);
   {---------------}
   { Current Speed }
   {---------------}
   v[2] := triang(0.384,0.768,2.560,CurrentVelocityStream);
   v[1] := radian(CartForm(v[1]));
   CalculateVector(v[1],v[2],current);
end;
```

2.6.4 VALIDATION OF THE CURRENT MODEL

The available current data does not lend itself to a specific empirical distribution, nor can it be approximated by some standard distribution. The lack of sufficient data to achieve this can be overcome by the methods used in 2.6.3. Although it can be argued that the method can produce results that are somewhat different from the actual conditions in the simulation area, the current model provides an input to the simulation that will affect the ship's movement and the missile's range tracking vector in a way that is more or less congruent with conditions on the South African seaboard.

If the effect of current on missile performance needs to be studied or, from a simulation point of view, a sensitivity analysis on this input needs to be done, these parameters can be adjusted accordingly.

2.7 MISCELLANEOUS MODELS

In this section we will look as the following less important models:

- Windage.
- Rain, Humidity and Oxygen.
- Air Temperature.
- Barometric Pressure.

2.7.1 WINDAGE

Windage is a term used for the influence of the wind on a ship's position. When a ship is stationary, the effect of windage is greatest and at maximum speed, the effect of windage is least. Also, the amount of windage experienced differs depending in the wind's relative direction to the ship. If the wind is abeam, windage will normally be at its greatest. However, trials that were conducted to establish windage factors for the WARRIOR-class FAC(M) of the SAN, showed that windage for that class of ship, because they are relatively small, was negligible. In general, FAC(M) is, for all practical purposes, much alike and thus we will discard windage as an environmental factor.

2.7.2 RAIN, HUMIDITY AND OXYGEN

Humidity and rain play an important part in the propagation of electronic emissions. For example, radar energy is attenuated by precipitation, water vapour and the oxygen in the air. When a portion of the radar energy incident on the molecules of water and oxygen is absorbed by it, the radar energy is attenuated and the radar range is reduced. Note[16] that the reduction in radar signal power when propagating over a distance R and back may be expressed as $\exp(-2\alpha R)$ where α is the one-way attenuation coefficient measured in units of the inverse distance. In the main, the attenuation of radar signals can be regarded as a statistical phenomena. This will be dealt with in subsequent chapters where the radar detection models are dealt with.

2.7.3 AIR TEMPERATURE

The prevailing air temperature influences, in part, the air density which, in turn, influences amongst other things, missile speed and the dispersion rate of chaff clouds. The South African Weather Bureau supplied the following information in regard to air temperature at the sea surface in the simulation area for the month of January:

- Mean Temperature in °C: $\bar{t} = 21.7$.

---

[16] M.I. Skolnik, *Introduction to Radar Systems*, 2 ed., Auckland: McGraw-Hill, 1981. p459.

- Temperature variance: $s_t^2 = 7.6$.

Furthermore, a normal distribution for all temperatures, whether it is day or night, is assumed. Although the variance is larger because we have not distinguished between day and night temperatures, this method obviates a procedure to simulate day or night with the resultant savings in computer run time. Variates representing air temperature in the simulation area will be generated by the polar method as described by Law and Kelton[17] and summarised in Algorithm 2.3.

Step 1:	Generate $U_1, U_2 \sim U(0,1)$ Set $V_1 = 2U_1 - 1$ $V_2 = 2U_2 - 1$ $W = V_1^2 + V_2^2$
Step 2:	If $W > 1$ the go to Step 1 Else Set $Y = \sqrt{(-2\ln W)/W}$ Return $X = V_1 Y$
Step 3:	Set $t = 21.7 + X\sqrt{7.6}$

Algorithm 2.3: Polar method for generating temperature.

2.7.4 BAROMETRIC PRESSURE

Barometric pressure is the other component that affects air density. However, the influence on missile speed is negligible[18] and as a result, barometric pressure will not be considered in the environmental model. See Section 4.3.2 for a complete overview.

[17] Law and Kelton. *op. cit.* p491.
[18] M.J. Zucrow and J.D. Hoffman., *Gas Dynamics Volume 1*. New York: John Wiley, 1976. P700.

Chapter 3

CONCEPT MODEL OF A FAST ATTACK CRAFT ARMED WITH MISSILES FAC(M)

3.1 PREMISE FOR SPECIFYING A GENERIC FAC(M)

In order to specify a generic FAC(M), we will first consider the range of FAC(M) deployed world wide. A histogram showing numbers of FAC(M) in various displacement ranges is in Figure 3.1.

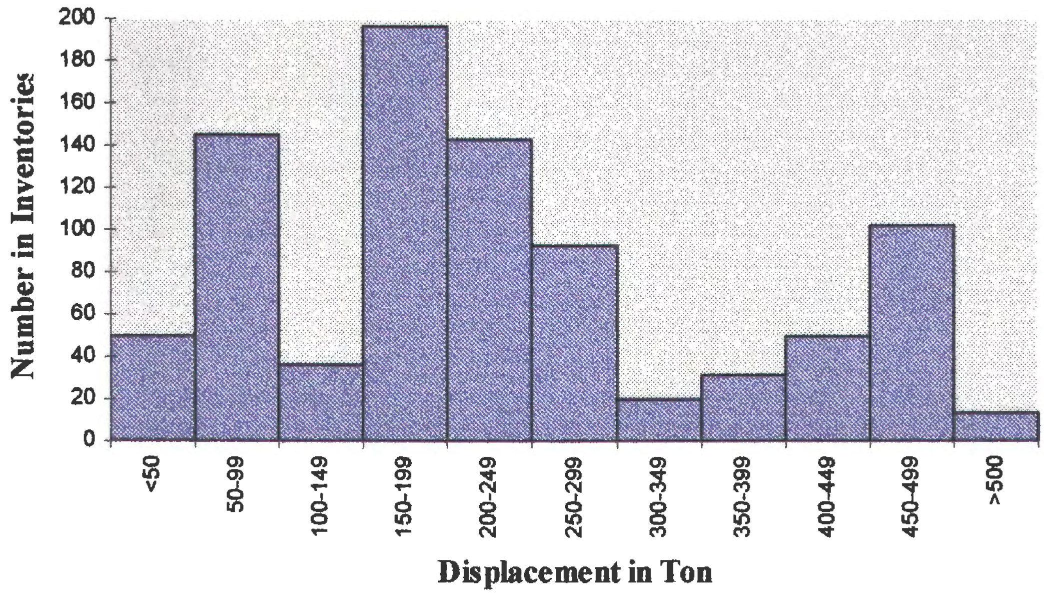


Figure 3.1: Displacement histogram of all FAC(M) in inventories world wide.

We note that three modes exist in the histogram above. The three modes are centred at about 75, 200 and 450 tons displacement respectively. These modes can be explained, in terms of their associated FAC(M) concept of operations, as follows:

- The mode at about 75 tons displacement includes FAC(M) that, in general, are armed with only two SSM, some guns, normally with a calibre smaller than 20mm, a missile targeting radar and with no electronic support measures equipment at all. The concept of operations for these small FAC(M) is that they are considered expendable. A navy employing these ships will regard them as one-shot fighting units. As a result, such a navy would require large numbers of these ships. The Red Chinese Navy, for example, has 75 Hegu-class FAC(M) at 68 tons displacement in their inventory whilst the Taiwanese Navy has 50 Hai Ou-class FAC(M) at 47 tons displacement in their inventory.

- The mode at about 200 tons displacement comprises somewhat larger FAC(M) that are normally armed with four or more SSM, some 30 to 40 mm guns that can be used to defend these ships against air attack by aircraft that will use ballistic weapons and to a lesser degree against oncoming SSM. They are generally an improvement on the smaller FAC(M) as their sea-keeping envelope and operational range allows them to be used in an extended manner to their smaller counterparts. Although they can defend themselves better than the smaller FAC(M), navies that rely on these are to a large extent forced to invest in large numbers of these ships. Again, Red China is the example. Her navy has one hundred OSA I FAC(M) in its inventory. The concept of operations are in essence the same as the concept of operations for the small ones. However, these ships can be used over larger areas and in more adverse weather conditions.

- The third mode at about 450 tons represents the more advanced types of FAC(M). These ships are normally armed with six or more SSM, 76 mm general purpose guns for AA, naval gunfire support and bombardment of the opposing force's (OPFOR) ships, navigation- and search radar, a full electronic support measures suite to counter oncoming missiles and to confuse the opposing force's tactical picture. This is made possible by their larger displacement and resultant larger spaces between decks. The concept of operations for these ships are similar to the concept of operations for larger ships such as corvettes and small frigates. These ships can fight independently but they generally lack the ability to engage submarines. They are normally used in packs of various sizes to form surface action groups to engage the OPFOR.

3.2 FORM AND DIMENSIONS

The ships that are represented by this third mode in the histogram fits the scenario described in Chapter 1. Thus, the generic FAC(M) defined for this study should fall in this category. As a result, we choose a generic FAC(M) with the following characteristics:

- Displacement: 450 tons.
- Length overall: 60 metres.
- Breadth: 8 metres.
- The basic upper deck layout for the ship is depicted in Figure 3.2.

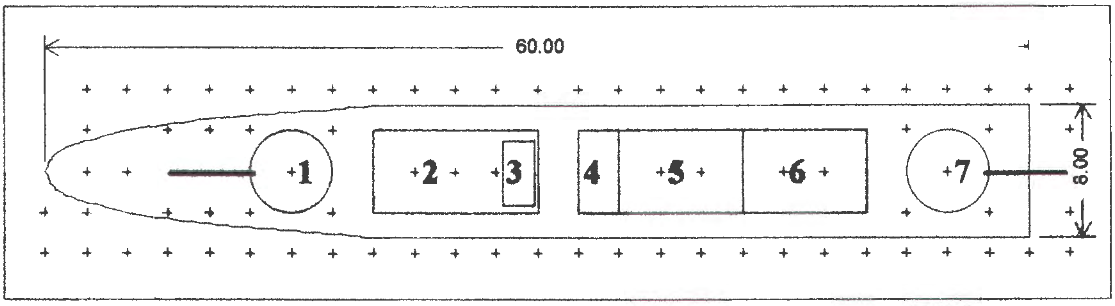


Figure 3.2: Basic dimensions and upper deck layout for generic FAC(M)

Note that the grid in Figure 3.2 represents 2.5 metres horizontally and vertically between marks and the numbers represents the following positions:

1. Forward 76mm gun.
2. Superstructure and Bridge.
3. Mast.
4. Engine Room Air Intake.
5. Forward Missile Position.
6. Aft Missile Position.
7. Aft 76mm gun.

3.3 RADAR CROSS SECTION

Skolnik[1] defines the *radar cross section* of a target as the fictional area intercepting that amount of power which, when scattered equally in all directions, produces an echo at the radar equal to that from the target or,

$$\begin{aligned}\sigma &= \frac{\text{power reflected toward source / unit solid angle}}{\text{incident power density} / 4\pi} \\ &= \lim_{R\to\infty} 4\pi R^2 \left|\frac{E_r}{E_i}\right|^2,\end{aligned} \tag{3.1}$$

where σ is the radar cross section (RCS) in square metres, R is the range between the radar and the target, E_r is the reflected field strength at the radar and E_i is the strength of the incident field at the target. For most common types of radar targets such as land mass, aircraft, missiles and ships, the radar cross section does not necessarily bear a simple relationship to the physical area except for that the larger the physical area, the larger the radar cross section is likely to be.

[1] M.I. Skolnik, *Introduction to Radar Systems*, 2 ed., Auckland: McGraw-Hill, 1981. p33.

In theory, the scattered field and therefore the radar cross section can be determined mathematically[2]. However, in practice this is only true of very simple shapes. A ship, for example, is a complex radar target and the only way to find her radar cross section is to measure it by some electronic method. The most common method is to illuminate a ship with a control radar under the proper conditions and to measure the returning echo strength by a calibrated receiver. The radar cross section is a combination of the radar cross sections of all the reflective surfaces of the ship. However, depending on the wavelength, some of these returning echoes may cause destructive interference, that is, they might cancel out one another. Thus, this combination of reflective surfaces is not necessarily linear nor can it be regarded, in a strict sense, as the sum of the returning echoes strengths. Also, the radar cross section will change as the target aspect changes[3]. However, for simplicity purposes in this simulation, the radar cross section will be assumed to be the sum of the radar cross sections from the various reflective surfaces for a given target aspect.

Furthermore, a ship may consist of a very large number of reflective surfaces. For example, in a typical mast alone, there may be forty or more objects such as legs, lattices and platforms. Also, there are a number of antennae of different shapes and sizes on the mast. In order to keep our generic FAC(M)s radar cross section calculations within reasonable bounds, only seventeen such surfaces were identified for the whole of the ship. They are depicted in Figure 3.3.

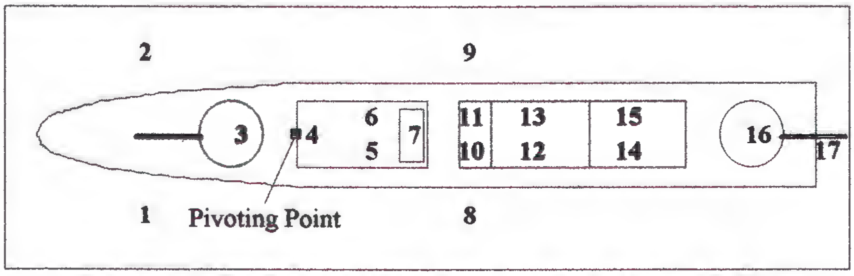


Figure 3.3: Reflective areas for the calculation of RCS.

The pivoting point in Figure 3.3 is the point around which the ship will rotate when altering course. The areas marked by numbers are explained in Table 3.1 below.

#	*Reflective Area*	#	*Reflective Area*
1	Port Bow	10	Air Intake - Port
2	Starboard Bow	11	Air Intake - Starboard
3	Forward 76mm Gun	12	Forward Missile Posn. - Port
4	Forward Superstructure Area	13	Forward Missile Posn. - Starboard
5	Port Superstructure Area	14	Aft Missile Posn. - Port
6	Starboard Superstructure Area	15	Aft Missile Posn. - Starboard
7	Mast and Rear Superstructure Area	16	Aft 76mm Gun
8	Port Ships Side	17	Stern
9	Starboard Ships Side		

Table 3.1: Explanation of numbered reflective areas in Figure 3.3.

[2] R.W.P. King and T.T. Wu, *The Scattering and Defraction of Waves*. Cambridge: Harvard University Press, 1959.
[3] Skolnik. *op.cit.* p 40.

Now, we have estimated the mean radar cross section for all seventeen surfaces for every 10° in relative bearing change. Thereafter all seventeen radar cross section estimates were summed to produce the overall radar cross section for the ship. The specific estimates for relative bearings on the port side and their sums are in Table 3.2 whilst the sums for all the relative bearings are presented graphically in Figure 3.4. Note that the radar cross section sums for starboard relative bearings mirror the ones on the port side.

	Relative Bearings on the Port Side																		
Posn	Stem	010	020	030	040	050	060	070	080	090	100	110	120	130	140	150	160	170	Stern
1	40	40	40	40	40	40	39	38	37	35	30	22	15	10	3	0	0	0	0
2	40	25	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	25	25	25	25	25	25	25	25	25	25	25	25	25	20	15	10	3	0	0
4	15	15	15	14	13	11	8	4	0	0	0	0	0	0	0	0	0	0	0
5	0	1	2	5	7	9	15	16	17	18	17	16	15	9	8	5	2	1	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	35	35	35	35	35	35	35	35	35	35	35	36	37	39	41	44	55	66	82
8	0	0	5	10	20	24	30	36	38	40	38	36	30	24	20	16	12	5	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	1	2	3	4	5	5	4	3	2	1	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	2	4	6	7	8	8	9	8	8	7	6	4	2	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	2	4	6	7	8	8	9	8	8	7	6	4	2	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	5	14	25	25	25	25	25	25	25	25	25	25
17	0	0	0	0	0	0	0	0	0	0	3	6	13	20	26	29	32	34	40
RCS	155	141	125	133	148	156	167	177	185	200	194	187	178	162	148	134	129	131	147

Table 3.2: RCS Estimates for the generic FAC(M).

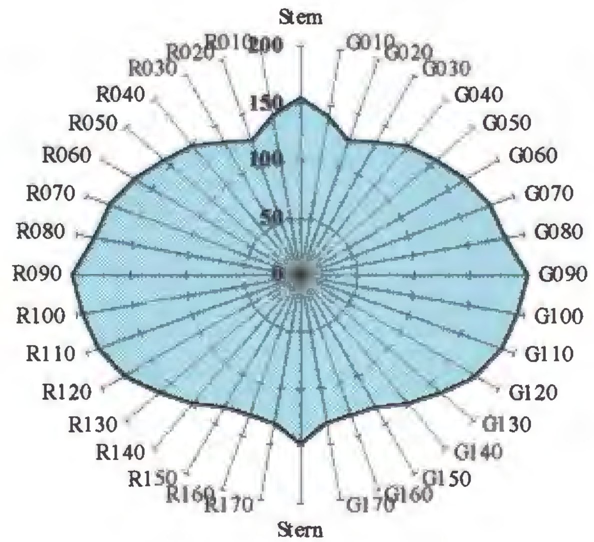


Figure 3.4: RCS plot for Generic FAC(M)

Also, we use the mean radar cross section values in the simulation because we make the assumption that the automatic gain control (AGC) loop of the missile head radar will ensure those values in the missile target tracking loop.

If we consider the application of stealth measures such as radar absorbent material, the radar cross section plot in Figure 3.4 adequately resembles the results found in real life situations for ships of similar design and size[4]. This set of data is assumed to be valid and sufficient for the simulation. In order to place the seventeen radar reflective areas in the correct positions, the areas are described in Table 3.3 in terms of their centre positions from the pivoting point and relative to the ship's head. Note that the longitudinal distances are referenced on the pivoting point and the athwartships' distances on the centre line. In other words, the positive longitudinal distances refer to a position on the centre line forward of the pivoting point and vice versa, whilst positive athwartships' distances lie to port of the centre line and vice versa.

Position	*Longitudinal*	*Athwartships*	*Position*	*Longitudinal*	*Athwartships*
1	+12	+3	10	-13	+2.5
2	+12	-3	11	-13	-2.5
3	+5	0	12	-18	+2.5
4	0	0	13	-18	-2.5
5	-5	+2.5	14	-26	+2.5
6	-5	-2.5	15	-26	-2.5
7	-8	0	16	-35	0
8	-20	+4	17	-40	0
9	-20	-4			

Table 3.3: Positional data in metres with respect to reflective areas relative to the ship's pivoting point.

The positional data is stored in file RCSPOS.DAT and the radar cross section data for various relative bearings is stored in file RCSMAG.DAT.

3.4 MANOEUVRING CHARACTERISTICS

In considering our generic FAC(M) and its manoeuvring characteristics, we shall do so in two parts, viz., firstly, we shall consider the initial position, course and speed and secondly we shall consider subsequent alterations in position, course and speed.

3.4.1 INITIAL SETTINGS FOR THE SIMULATION

Remember, from Section 2.4, that we will follow the convention that the ship's initial position for every replication of the simulation will be the position (0,0) in our Cartesian grid. Thus, we will simply set the ship's position accordingly at the start of every replication.

As it is important for our simulation that we consider all possible combinations of the relative positions of the ship to the missile, we choose the ship's initial course

[4] P.L. Botha, *Toepassing van Radar Absorberende Materiaal vir Vermindering van die Aanvalsvaartuig Radardeursnit.* Simon's Town: IMT, October 1986. pp 22-23.

randomly. Therefore, we assume that the ship's initial course is uniformly distributed between 0°T and 360°T. However, this will result in a uniform distribution of the ship's initial course in the Cartesian grid as well. As a result, we generate the ship's initial course from the latter distribution directly. Also, ships are normally steered to the nearest full degree. Therefore, the ship's initial course is rounded to the nearest full degree.

Economical speed for a typical vessel of 450 tons displacement, powered by four marine diesel engines, each connected to its own propeller, is in the region of 22 knots. The typical maximum sustainable speed for such a vessel is about 30 knots[5]. However, from practical experience at sea, officers commanding such ships invariably elect to restrict their vessel's speed to at least two knots less than maximum sustainable speed. Therefore, we will consider the initial speed interval [22,28] only. In the absence of any indicators to the contrary, we choose a uniform distribution in the considered speed range to generate the ship's initial speed.

3.4.2 CHANGES IN THE SHIP'S COURSE

A ship's *turning circle* is the path followed by the pivoting point of that ship in making a turn of 360° or more at a constant rudder angle and speed. The pivoting point is typically about one-third the way aft from the bow[6]. This may vary from one vessel to the other and it may also vary for a given vessel under different conditions of longitudinal trim. The stem will turn on the inside of the turning circle and the stern on the outside of the turning circle.

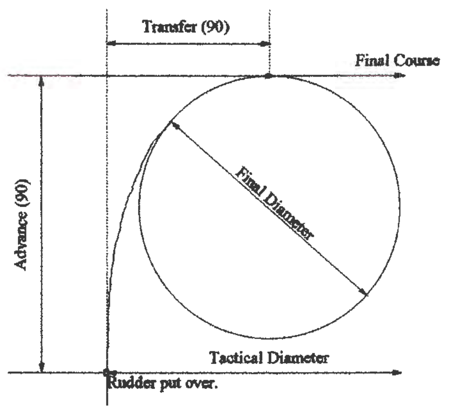


Figure 3.5: Advance, Transfer and Diameters.

We define[7] *advance* as the distance gained in the original direction until the ship steadies on her new course. Similarly, *transfer* is the distance gained at right angles to the original course measured from the line representing the original direction of travel to the point where the ship steadies on her new course. The advance and transfer for a ship altering her course 90° to starboard is shown in Figure 3.5.

---

[5] *Jane's Weapon Systems 1980/81* edited by R.T. Pretty. London : Jane's Yearbooks,[1980].
[6] E.S. Maloney, *Dutton's Navigation and Piloting*, 13 ed., Annapolis: Naval Institute Press. 1978. p300.
[7] *Ibid.* p301.

Now, we observe that the turning circle for any vessel will not be constant. The rudder angle applied and the speed at which the ship will be steaming will have the following effects on the turning circle:

- The larger the applied rudder angle is, the smaller the turning circle will be and vice versa.

- The higher the ship's speed is, the larger the turning circle will be and vice versa.

From practical experience, in a tactical scenario, the rule of thumb is to use maximum rudder to facilitate a rapid reaction to the tactical needs that prevail. We assume that when alterations of course are necessary for countering an oncoming missile, rapid reaction is important.

From Figure 3.5, it is clear that, initially, the alteration of course is not constant. This can be explained by saying that because the athwartships' forces induced on the ship by the applied rudder angle must first overcome the ship's momentum on the previous course. As a result, the rate of angular change in the ship's course is initially small. Thereafter, it increases until the athwartships' forces induced on the ship by the applied rudder angle are congruent with the ship's angular movement around the pivoting point where the rate of angular change in the ship's course will stabilise. Furthermore, this will explain the difference between the tactical diameter and the final diameter in Figure 3.5.

	Time Elapsed in Seconds								
Speed	5	10	15	20	25	30	35	40	45
22	1.0	2.0	3.0	4.0	5.0	6.0	7.0	7.0	7.0
24	1.1	2.2	3.3	4.5	5.6	6.7	7.5	7.9	8.0
26	1.2	2.4	3.5	4.7	6.0	7.0	8.0	8.9	9.0
28	1.3	2.7	3.8	5.0	6.2	7.3	8.5	9.0	9.0
30	1.5	2.9	4.0	5.3	6.5	7.6	9.0	10.0	10.0

Table 3.4: Angular rate of course change in degrees per second for various speeds in knots.

In order to model the alterations of course and the resultant advance and transfer for our generic FAC(M), we shall consider the angular rate of change from the time the rudder is put over until the ship has steadied on its new course for various speeds. However, we shall consider speeds in the interval [22,30] only as we assume that the initial speed will not be less than 22 knots and the maximum speed of our generic FAC(M) is 30 knots. To that end, the angular rates for changes in course with maximum rudder applied and with various speeds rung on, were experimentally determined for the WARRIOR-class FAC(M) of the South African Navy. Similar rates of change were assumed for our generic FAC(M) and are contained in Table 3.4. Note that the rates of change are symmetric for alterations of course to port and for alterations of course to starboard.

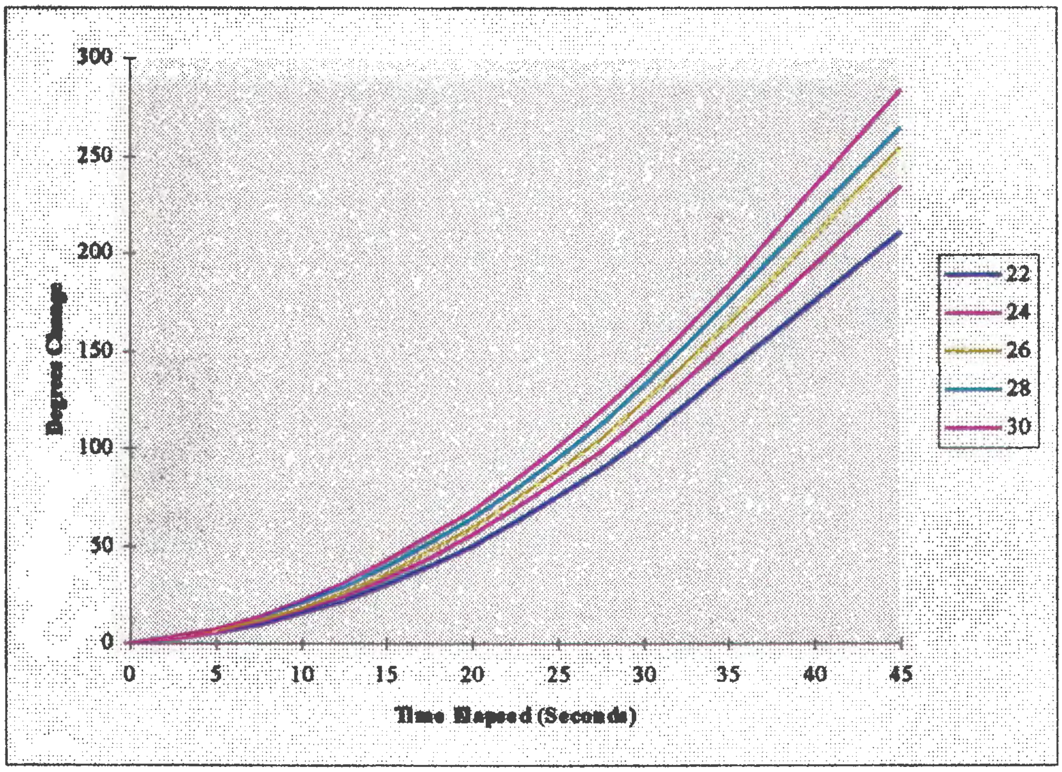


Figure 3.6: Degrees of course change in regard to time with maximum rudder applied.

Note that the data in Table 3.4 is contained in file ALTCO.DAT. Also, Figure 3.6 gives a representation of how much the ship's course will change in regard to time for various speeds rung on and with maximum rudder applied.

3.4.3 CHANGES IN SHIP'S SPEED

From Table 3.4 and Figure 3.6, it can clearly be seen that the ship is more responsive in regard to altering course at higher speeds. Therefore, in the light of our discussion in Section 1.6, it can be argued that it might be an optimal course of action to ring on as high a speed as possible when being engaged by missiles in order to manoeuvre quickly.

A ship, similar to our generic FAC(M) dimensions, propulsion and displacement, normally can accelerate from 22 knots to 30 knots in about 20 seconds. Typical acceleration curves from WARRIOR-class FAC(M) would indicate that the curves in Figure 3.7 are sufficient for our purpose. From these curves, albeit somewhat course, we can deduce that acceleration in speed for such vessels is roughly 0.4 knots per second. We assume this figure is necessary and sufficient for the chosen generic FAC(M).

When a ship de-accelerates, it can be argued that the rate of deceleration will be greater than 0.4 as friction and other factors will work in on it to slow the vessel down much faster than that rate. As we will not consider this situation in the simulation, the matter will be excluded from further consideration.

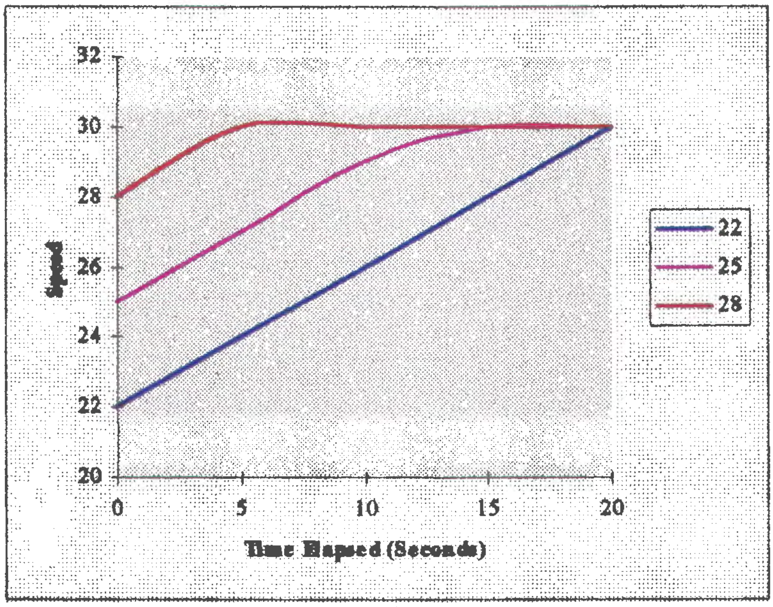


Figure 3.7: Acceleration Curves.

3.4.4 REACTION TIME

In the event that a course or speed change is ordered, there is some time delay from when the reason for the course or speed change becomes apparent to when the required action is taken, for example, there is a time delay from when an oncoming missile is detected until the required wheel order and throttle order have been implemented.

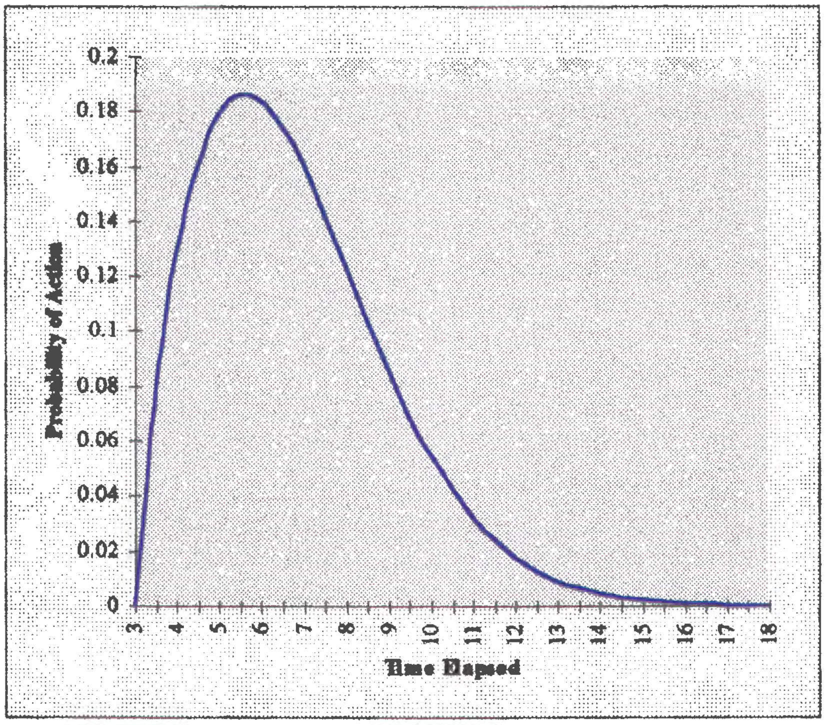


Figure 3.8: $r_t = 3 + W$ and $W \sim$ Weibull(1.75,4.25).

From observing the normal drill and procedure aboard a WARRIOR-class FAC(M), it is concluded that the reaction time described above, r_t, has a Weibull distribution such that

$$r_t = 3 + W \qquad (3.2)$$

where $W \sim$ Weibull(1.75,4.25). The mass function for this distribution is in Figure 3.8. Also, note that although there were only 25 observations, which is relatively few, it was decided to use the Weibull distribution rather than estimating a distribution by some other distribution such as the triangular distribution which might be regarded as more applicable when data is absent or only a small sample is available. This was done because experience[8] has shown that the Weibull distribution is more representative in such cases.

3.4.5 MANOEUVRING MODEL

When it becomes necessary for the FAC(M) to manoeuvre, the Pascal source code below, will be employed.

```
if (Acourse OR Aspeed) AND ReactionTimeElapsed then
   begin
      if abs(NewSpeed-ship.speed)>SpeedDiff then
         begin
            ship.speed := ship.speed + SRate/prf;
         end
      else
         begin
            Aspeed     := FALSE;
            ship.speed := NewSpeed;
         end;
      if abs(NewCourse-ship.head)>CourseDiff then
         begin
            if TurnPort then
               begin
                  ship.head := ship.head + turnrate/prf;
                  if ship.head>(2*pi) then
                     begin
                        ship.head := ship.head - 2*pi;
                     end;
               end
            else
               begin
                  ship.head := ship.head - turnrate/prf;
                  if ship.head<0 then
                     begin
                        ship.head := ship.head + 2*pi;
                     end;
               end;
         end
      else
         begin
            Acourse   := FALSE;
            ship.head := NewCourse;
         end;
      CalculateVector(ship.head,ship.speed,ship.move);
```

[8] A.M. Law, and W.D.Kelton, *Simulation Modeling and Analysis* (*sic.*). 2 ed. New York: McGraw-Hill, 1991, p 333.

```
    ShipRCSPlan;
end;
```

Note that `Acourse`, `Aspeed` and `ReactionTimeElapsed` are Boolean variables that will enable changes in course and speed in the simulation.

3.5 RADAR DETECTION OF AN ONCOMING SSM

The detection of the oncoming missile is dependent on the radar equation. Now, consider a radar with an isomorphic antenna, that is, an antenna that transmits in all directions. The area of the sphere in which the antenna transmits at a given range R is $4\pi R^2$. The power density of the radar pulse, P_t, transmitted by the radar at any given point at range R will therefore be

$$\frac{P_t}{4\pi R^2}.$$

If we replace the isomorphic antenna by a directional antenna with a gain of G, then the power density of the radar pulse transmitted by the radar at any given point at range R will therefore be

$$\frac{P_t G}{4\pi R^2}.$$

Furthermore, the target intercepts a portion of the incident power and re-radiates it in various directions. The measure of the amount of incident power intercepted by the target (See 3.1) and re-radiated back to the receiving antenna is thus defined as:

$$\text{Power density of echo signal at the radar receiving antenna} = \frac{P_t G}{4\pi R^2}\frac{\sigma}{4\pi R^2}.$$

If we take the size of the antenna, A_e, into account, then the power density of the echo signal received by the antenna is

$$\frac{P_t G}{4\pi R^2}\frac{A_e \sigma}{4\pi R^2}.$$

By extracting R from this equation and allowing for the minimum detectable signal we obtain the radar equation which predicts the maximum search radar detection range of a particular radar target for a particular radar. The maximum detection range, R_{max}, in metres is thus given in its simplest form by

$$R_{max} = \sqrt[4]{\frac{P_t G A_e \sigma}{(4\pi)^2 S_{min}}} \qquad (3.3)$$

where P_t is the transmitted power in watts, G is the antenna gain as a ratio, A_e is the antenna effective aperture in m^2, σ is the radar cross section in m^2 and S_{min} is the minimum detectable signal in watts[9].

However, in practice this simple radar equation does not predict the detection range for radar targets with sufficient accuracy. Both S_{min} and σ are statistical in nature and must be expressed in that manner. Now, there is a fair amount of self-generated noise present in the radar receiver (Rx). To detect radar targets, a threshold is set that will inhibit the receiver from seeing this self-generated noise and as a result, returning echoes which are smaller than the set threshold. Because of temperature, humidity and other conditions this threshold may vary from time to time. Therefore, S_{min} will also vary statistically.

The statistical nature of σ lies in the fact that the target is an object that normally moves in three dimensions. Small changes in the target aspect influences σ significantly. In turn, that influences the signal strength of the returning echo. Also, the noise generated in the radar target environment, for example, the sea close to a missile being illuminated by radar (sea clutter), may inhibit the detection of the radar target by the radar. There exists a ratio between the target generated echo and the echoes generated by its environment. A signal-to-noise ratio, similar to the receiver threshold must be set in the radar's intermediate frequency section in order to rid the radar picture of false echoes.

Furthermore, for radar targets close to the sea surface, radar energy that is reflected from the sea onto the target also creates return echoes which, in turn, may enhance or reduce the received returning echo strength (multi-path effect) because the indirectly generated returning echoes are of a different phase to the directly generated echoes.

Also, (3.3) assumes that the radar transmits in free space, that is, the radar transmits in a vacuum. That means that (3.3) assumes there are no losses in the returning echoes from the radar target due to energy absorption in the atmosphere. To allow for the losses and propagation effects and because A_e can be considered as part of the gain when only one antenna is used, we modify (3.3) as follows[10]:

$$R_{max} = \sqrt[4]{\frac{P_t G^2 \lambda^2 \sigma}{(4\pi)^3 S_{min} L}}, \qquad (3.4)$$

where λ is the wavelength in metres and L is all the losses due to external and internal noise. Note that by antenna theory

$$G = 4\pi A_e / \lambda^2$$

and that the losses is normally expressed in dB and that the conversion for x dB from dB to linear units in (3.4) is the anti-logarithm $L_{(3.4)} = 10^{x/10}$

[9] Skolnik. *op. cit.* p 15.
[10] *Ibid.*

Note that, although Skolnik[11] indicates that most of the randomness in (3.4) can be adequately described by the normal distribution, little or no generalisation of the distributions of S_{min} and σ are possible because we cannot experiment sufficiently for our chosen radar and the incoming missile is generic in nature. In order to predict, given a particular radar, at what ranges a particular radar target can be detected and what the probability of detection at that range is, the Institute for Maritime Technology (IMT) has developed the Justice Radar Simulation[12]. This model was used extensively to investigate the radar detection problem. Also, the results were used to decide on the radar detection model for our simulation.

3.5.1 GENERIC SEARCH RADAR

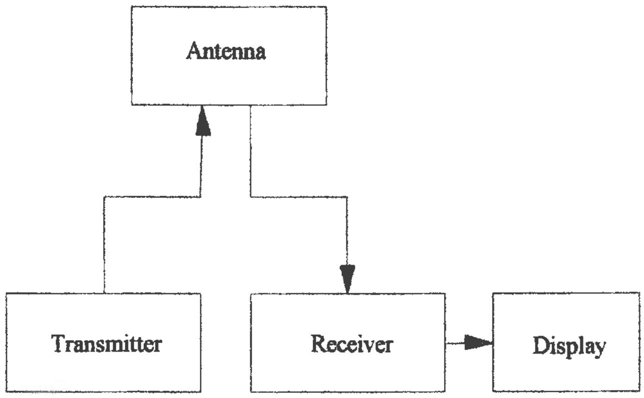


Figure 3.9: Simple Search Radar Block Diagram

As the classic search radar operates in the lower radar frequency bands, we choose our generic search radar to operate in the F-band and, furthermore, we choose the parameters set out below.

Radar Type	:	Search Radar : F-band.
Transmitter		
Wavelength		: 10 cm.
Peak Power		: 450 kW.
Pulse Length		: 4 μs.
PRF		: 500 hz.
Tx Loss		: 2 dB.
Receiver		
Noise Level		: 6 dB.
System Losses		: 9 dB.
Rx Losses		: 5 dB.
Pulses Integrated		: 32.
Integration		: Coherent.

[11] Skolnik. *op. cit.* pp 23 - 27.
[12] D.A. Harrison., *Radar Based Weapons System Simulation Model*. Simon's Town : IMT. Nov 1980.

Antenna

Gain	: 34 dB.
Vertical Beam Width	: 8.3°.
Horizontal Beam Width	: 1.2°.
Scan Rate	: 12 r.p.m.
Side Lobe Level	: -30 dB.
Height	: 17 m.
Polarisation	: Vertical.

Discrimination

Type	: Signal Processing.
Processing Losses	: 2 dB.

Errors

Imp Height Error	: 3.5 m.
Timing Jitter	: 15 ns.
Height S/N Error	: 10 m.
Bearing S/N Error	: 10 mr.
Elevation S/N Error	: 10 mr.

3.5.2 EXTERNAL LOSSES IN THE RADAR EQUATION

Beaufort Number	Seaman's Description	General Rules	Wind Velocity (knots)
0	Calm	Water surface smooth and glassy.	0-1
1	Light Air	Parts of water surface wind ruffled with smooth patches interspersed.	1-3
2	Light Breeze	All surfaces ruffled.	4-6
3	Gentle Breeze	Small waves with occasional white caps.	7-10
4	Moderate Breeze	About half the wave tops breaking.	11-16
5	Fresh Breeze	Entire surface broken into white caps.	17-21
6	Strong Breeze	Tops of wave blowing off; spume.	22-27
7	Moderate Gale	High waves breaking on crests.	28-33
8	Fresh Gale	High waves breaking on crests.	34-40
9	Strong Gale	Very high waves breaking on crests.	41-47
10	Whole gale	Extremely high waves breaking on crests.	48-55
11	Storm	Extremely high waves breaking on crests.	56-65
12	Hurricane	Extremely high waves with severe breaking on crests.	Above 65

Figure 3.10: Beaufort Scale

The major contributor to external losses is the fact that the radar does not operate in free space, that is, the radar is operating over the sea and as a result, the propagation of the radar energy is dissipated by it. Furthermore, some of the dissipated energy may even return to the radar in its side lobes causing false and often spurious echoes on the radar display. The extent to which the sea influences these losses is dependent on the prevailing sea state. In fact, there is a strong positive correlation between the sea state and the dissipation of the radar energy to and from the radar target. Seafarers describe the sea state in accordance with the Beaufort scale[13]. The Beaufort scale is depicted in Figure 3.10.

[13] C.D. Lane and J.D. Sleightholme, *The New Boatman's Manual*. Adlard Coles Ltd. 1967. p 241.

By inspection of Figure 3.10, we note that sea states 0 to 7 are applicable to our simulation. Therefore, we will restrict further research to radar energy losses due to the sea to these sea states. Remember from Section 2.3 that we will choose the generic SSM so that it can fly over wave heights at the 95th percentile, that is, over waves of 8.5 metres or less.

3.5.3 JUSTICE SIMULATION OF SEARCH RADAR DETECTION

Figure 3.11 depicts the prediction of search radar detection by the Justice radar simulation for sea state 0 and Figure 3.12 depicts same for sea state 7. The complete results of the various simulations are in Appendix D.

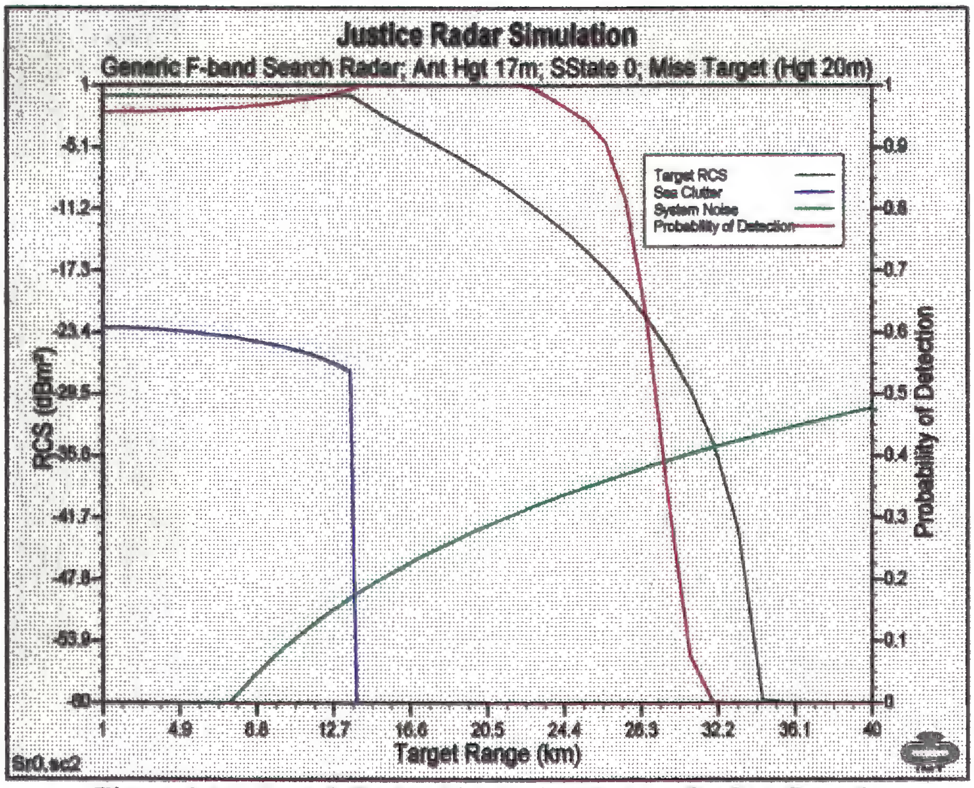


Figure 3.11: Search Radar Simulation Output for Sea State 0.

From Figure 3.11 we note that where the sea clutter noise drops off due to the sea horizon, the probability of detection is best. Furthermore, the probability of detecting an oncoming missile increases rapidly as it comes over the missile height horizon which is at a greater range than the sea horizon. As the missile closes the radar, the probability of detection decreases due to the losses incurred by the sea clutter. From the graph, it is easy to see that the probability of detection reduces from 100% just beyond the sea horizon to about 95% one kilometre from the radar.

From Figure 3.12 we see that, again the probability of detecting an oncoming missile increases rapidly as it comes over the missile height horizon until it becomes best were the sea clutter noise drops off due to the sea horizon. Also, the probability of detection reduces rapidly from 100% just beyond the sea horizon to zero at about 16 kilometres from the radar. This is due to the large sea clutter in sea state 7.

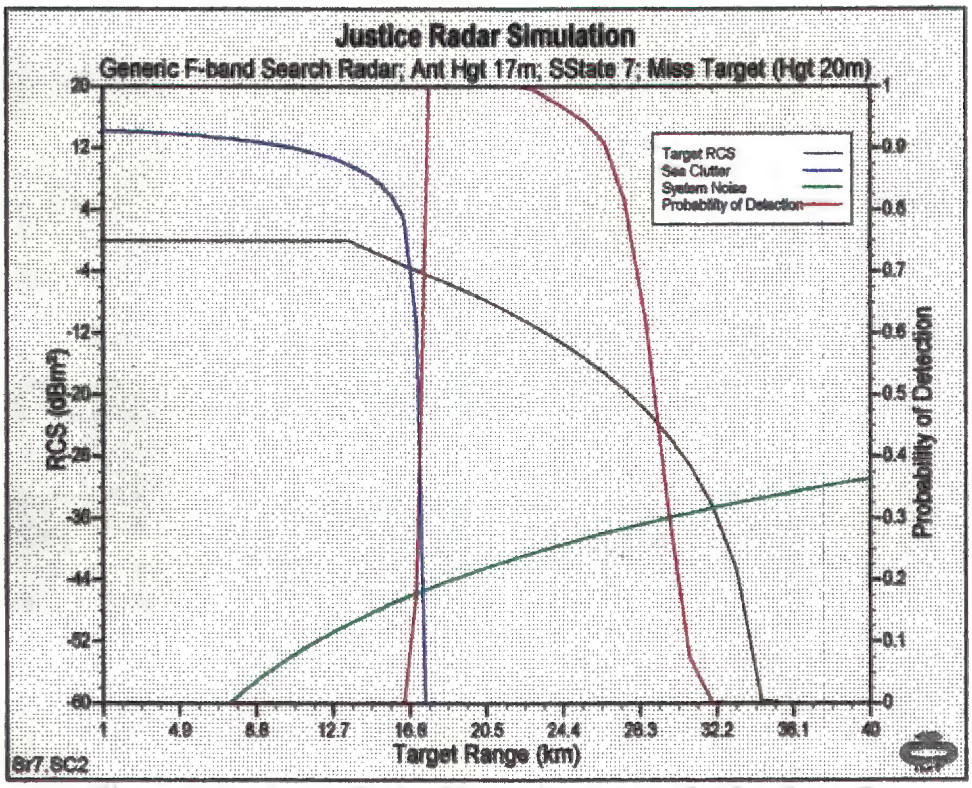


Figure 3.12: Search Radar Simulation Output for Sea State 7.

Sea State	Detection Probability (Beyond Sea Horizon)				
	0.00	0.25	0.50	0.75	1.00
0	32.68	29.97	28.96	27.81	22.21
1	32.50	30.14	29.12	27.81	22.33
2	32.29	29.97	28.78	27.96	22.21
3	32.29	29.97	28.96	27.81	22.33
4	31.93	29.97	28.96	27.81	22.47
5	32.29	29.97	29.12	27.96	22.33
6	32.12	29.97	28.96	27.81	22.33
7	32.50	29.97	28.96	27.81	22.47

Table 3.5: Detection Probability for various sea states and ranges in kilometres.

We deduce that although the sea clutter in the two sea states respectively have a significantly different impact on the probability of detecting the oncoming SSM inside the sea horizon, it may well have no impact on the probability of detecting the oncoming SSM over the sea horizon but inside the missile height horizon. The Justice simulation model was revisited for all sea states and the probability of detecting the missile in the latter area is summarised in Table 3.5.

From Table 3.5, we see that as there is extremely little difference in the distances associated with particular detection probabilities, our deduction holds true for all simulated sea states, that is, although the sea clutter in the various sea states have a significantly different impact on the probability of detecting the oncoming SSM inside the sea horizon, it may well have no impact on the probability of detecting an oncoming SSM over the sea horizon but inside the missile height horizon for sea states 0 to 7.

3.5.4 RADAR DETECTION MODEL

In order to decide on a radar detection model for our simulation, we assume that if the radar detects an oncoming missile, the radar operator will detect it also. This means that we assume that the operator will see an oncoming missile on the radar display given that the radar has detected that missile, or,

$$P(\text{Operator see SSM}|\text{Radar detect SSM}) = 1.$$

Furthermore, if we assume that there is some automatic warning device that will sound a warning if the search radar detects an oncoming missile, that is, a radar target that exceeds some pre-set velocity, then the assumption above is realistic. Also, such a device is possible if the search radar is equipped with a Doppler or moving target indication (MTI) receiver.

Table 3.5 indicates that there are no real differences in the ranges for particular probabilities. Hence, we choose a simple radar detection model that incorporates the average ranges from the table for the particular probabilities. This is summarised in Table 3.6 and graphically illustrated in Figure 3.13.

P(Radar detect SSM)	0	0.25	0.5	0.75	1
Detection Range	32.325	29.99125	28.9775	27.8475	22.335

Table 3.6: Search radar detection distribution

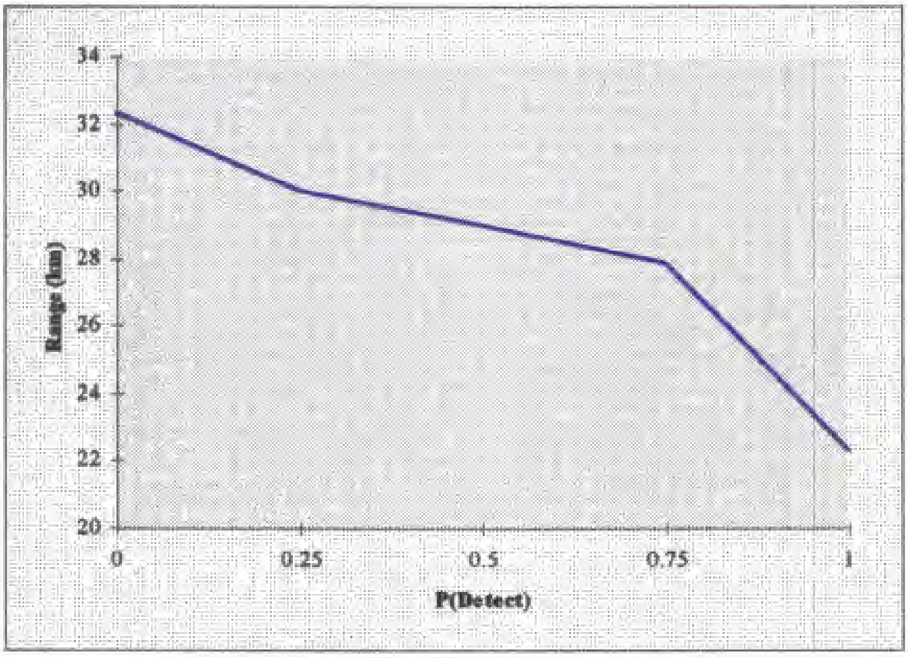


Figure 3.9: Search radar detection distribution

In order to decide on the search radar detection range we consider it necessary and sufficient to generate $U \sim$ U(0,1) and to use U to predict the detection range similarly to what we did to generate wind speed in Algorithm 2.2. The Pascal source code for finding the radar range for a particular replication of the simulation follows directly.

```
function DetectionRange(R : radartype):real;
{------------------------------}
{ Return Radar Detection Range }
{------------------------------}
var U,
    P : real;

begin
   U              := rand(DetectRadarStream);
   P              := 4 * U;
   J              := trunc(P) + 1;
   DetectionRange := R[J] + (P - J + 1)*(R[J+1] - R[J]);
end;
```

3.6 ELECTRONIC WARFARE EQUIPMENT

We will consider the following four electronic warfare machines:

- A passive radar receiver which supplies, provided the MHR is active, an early warning and bearing of the oncoming missile.

- A spot jammer that is able to jam the oncoming missile. By *jamming* is meant the transmission of radar energy to obscure the ship's radar cross section in the missile head radar's receiver by generating sufficient radar energy in a very narrow frequency band.

- Close range chaff that will generate alternative radar reflective targets close to the ship.

- Medium range chaff that will generate alternative radar reflective targets in the missile's search path.

Note that we will not consider range gate stealers because, in general, it is considered that their reaction time is such that they cannot produce radar energy in the missile's early tracking gate, nor are the present travelling wave tube (TWT) technology able to place sufficient energy in the missile's late tracking gate to affect the missile's tracking algorithm.

3.6.1 DETECTION OF THE MISSILE HEAD RADAR (MHR)

In order to give an earlier warning of an oncoming missile than what can be expected from the search radar, the ship uses a passive radar receiver which detects the missile head radar transmissions and finds the bearing of the transmission source. This type of equipment is normally referred to as an Electronic Intelligence (Elint) receiver and it forms part of the so-called electronic support measures (ESM).

The bearing accuracy of an Elint receiver depends on factors such as the antenna width, bandwidth of the receiver and signal strength experienced. Typical bearing accuracy for a good Elint receiver is two degrees. For a lesser Elint receiver the bearing accuracy may be as large as ten degrees. We assume a good Elint receiver for our simulation. Thus, we assume the bearing, b_{ESM}, in degrees, produced by the Elint receiver of an oncoming missile is uniformly distributed such that

$$b_{ESM} \sim U(b-2, b+2)$$

where b is the actual bearing in true degrees of the radar energy source. The Pascal source code to find b_{ESM}, denoted `ESMbrg` in the program, follows directly.

```
function ESMbrg : real;
{------------------------------------------------------------}
{ Returns the ESM bearing of an active radar source detected }
{ x represents east and y represents north co-ordinates      }
{------------------------------------------------------------}
var b,
    x,
    y : real;

begin
   x := SSM.posn[1] - ship.posn[1];
   y := SSM.posn[2] - ship.posn[2];
   if x<0 then
      begin
         b := pi + arctan(y/x);
      end
   else
      begin
         if y<0 then
            begin
               b := 2*pi + arctan(y/x);
            end
         else
            begin
               b := arctan(y/x);
            end;
      end;
   b := degree(b);
   ESMbrg := trunc(uniform(b-2,b+2,ESMbrgStream));
end;
```

However, electronic support measure equipment has limits on how far it can detect radar energy sources. From the Justice simulation, we see that the maximum radar target horizon, for a missile at 20 metres and a search radar at 17 metres, is 35.125 kilometres whilst the 100% maximum detection range of the search radar is 32.68 kilometres. From own experience, a good combat operator rule of thumb is that electronic support measure equipment will detect a radar at about 1,4 times the detection range of the radar under consideration, that is, the electronic support measure advantage over radar is about 1.4:1. However, the electronic support measure detection range is not constant and by the central limit theorem we assume that it is normally distributed. However, we do not know the standard deviation of the detection range. In the absence of any data, we assume the variance of the electronic support measure detection range to be equal to half of the radar detection range, or

$$R_{\text{ESM}} \sim \text{normal}(R_{\text{SR}}, R_{\text{SR}}/2).$$

However, if R_{ESM} is larger than the distance to the radar horizon, it is reasonable to expect that R_{ESM} will be limited to the maximum radar target horizon, R_{HORIZON}. Also, from practical experience, this seems to be the case for about ninety percent of the time. Now, in order to model the electronic support measure detection range, we choose R_{ESM} such that

$$R'_{\text{ESM}} = \min\left(R_{\text{ESM}} \sim \text{normal}(R_{\text{SR}}, R_{\text{SR}}/2), R_{\text{HORIZON}}\right)$$

By experimenting with the simulation model we note that the result of choosing this model for the first hundred replications is that R'_{ESM} is less than the maximum radar target horizon for 13% of the replications. A 95% confidence interval for $R_{\text{ESM}} < R_{\text{HORIZON}}$ is 0.13 ± 0.066. We deduce that the simulation model output is not significantly different from the ten percent assumed by experience. We conclude that the electronic support measure range generator is sufficient.

3.6.2 SPOT-JAMMER

In the most general sense, a *jammer* is an Electronic Counter Measure (ECM) device that transmits radar energy, either across a part of or across the whole bandwidth of a victim electronic system[14]. The output of a jammer, that is, the amplitude of the noise generated over a particular bandwidth is generally distributed normally. The probability density function for the amplitude of a jammer output versus the frequency generated is depicted in Figure 3.14.

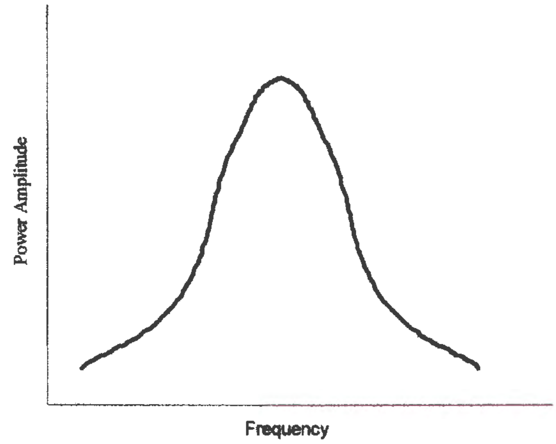


Figure 3.14: Amplitude vs Frequency Graph for a Typical Jammer[15].

Now, a *spot-jammer* is a jammer that transmits radar energy over such a small bandwidth that, for all practical purposes, the jammer can be considered as transmitting on a particular frequency. We choose our jamming device to be a spot-jammer.

---

[14] L. B. Van Brunt, *Applied ECM - Vol 1*. Dunn Loring, Va.: EW Engineering, c1978. p 293.
[15] *Ibid.* p 297.

Furthermore, we require that the spot-jammer will cover the missile head radar frequency bandwidth adequately, that is, the spot-jammer will mask the radar reflections from the ship caused by the missile head radar to such an extent that the missile head radar receiver will not be able to discern the ship's radar reflections from the jammer noise. However, note that a *burn-through range*, that is, a radar to target slant range where the radar receiver can detect the skin reflections of an oncoming target, such as a ship, previously obscured by jamming[16], exists.

We know[17] that in free space the returned power density from a radar target, I_r, can be expressed as

$$I_r = \frac{P_t G A_t}{(4\pi)^2 R^4} \tag{3.5}$$

where A_t is the target area.

Furthermore, if the jammer transmits with gain, G_j, then the one way power density from the jammer, I_j, can be expressed as

$$I_j = \frac{P_j G_j}{4\pi R^2} \tag{3.6}$$

where j denotes the jammer. Since the power densities, also called *intensities*, decrease with range at unequal rates, larger ranges favour the jammer whereas shorter ranges favour the radar. Therefore, there is a unique range, the burn-through range or self-screening range, inside of which, the radar prevails. The self-screening range, R_{ss}, may be found by equating (3.5) and (3.6) and solving for R_{ss} by

$$R_{ss} = \sqrt{\frac{P_t G A_t}{P_j 4\pi}}\,. \tag{3.7}$$

If the jammer has a directive gain capability, as our simulation assumes, then R_{ss} would also be found by setting the two intensities equal, or

$$I_j = I_r,$$
$$\frac{P_j G_j}{4\pi R_{ss}^2} = \frac{P_t G_t A_t}{(4\pi)^2 R_{ss}^4},$$

which yields

$$R_{ss} = \sqrt{\frac{P_t G_t A_t}{P_j G_j 4\pi}} \text{ m.} \tag{3.8}$$

---

[16] Van Brunt. *op. cit.* p 41.
[17] *Naval Operations Analysis*. 2 ed. Annapolis, Ma: Naval Institute Press. 1977. pp 96-98.

Now, if we take the missile head radar parameters for our generic SSM in Appendix D and the radar cross section data in Figure 3.4, appertaining to our generic FAC(M), into account and we specify the parameters in Table 3.7 for our spot-jammer, then by (3.8) the FAC(M)'s self-screening range is

$$R_{ss} \approx 2617 \text{ metres}$$

for the FAC(M) maximum mean radar cross section of 200 m^2 and

$$R_{ss} \approx 2067 \text{ metres}$$

for the FAC(M) minimum mean radar cross section of 125 m^2.

P_j	5 kW
G_j	32 dB

Table 3.7: Jammer parameters

We choose the spot-jammer parameters as tabled in Table 3.6. the implication is that, in general, our jammer will not screen the ship from the missile head radar for ranges less than 2617 metres. Thus, a tactic whereby the oncoming missile is jammed when its range from the ship is less than 2617 metres cannot be entertained. As a result, the jammer model can be simplified to a Boolean value such that when the missile is further away than R_{ss} then

```
JAMMER := TRUE;
```

or

```
JAMMER := FALSE;
```

and when the missile is closer than R_{ss}, then

```
JAMMER := FALSE;
```

If the jammer is on, that is `JAMMER := TRUE`, the effect on the missile head radar will be a single radar source based on the ship's mast position (−8,0) as the jammer will screen the rest of the ship effectively. Furthermore, any other radar reflective surfaces such as chaff will be discounted as their radar cross section will also be screened effectively[18], provided the deployed chaff clouds are sufficiently close to the ship and therefore need not be modelled whilst the jammer is transmitting at the missile head radar.

3.6.3 CLOSE RANGE CHAFF

We note that the Oxford Dictionary defines chaff as the husks of grain or anything useless. However, a modern day electronic warfare definition[19] defines *chaff* as elemental passive reflectors, absorbers or refractors of radar, communications and other weapon system radiations which can be floated or otherwise suspended in the atmosphere or exo-atmosphere for the purpose of confusing screening or otherwise

[18] *Naval Operations Analysis. op. cit.* p 99.
[19] Van Brunt, *op. cit.* p 377.

adversely effecting the performance of victim electronic systems. From this definition, we see that chaff can be classified as an electronic countermeasure.

Close range chaff (CRC) consists of small radar reflective dipoles that reflect radar energy. The dipoles are cut to half the wavelength of the radar that it is supposed to victimise. In our case, the generic close range chaff dipoles will be cut to 0.5 cm as that is half the wavelength of the missile head radar. Note that chaff is packed in bundles consisting of a certain amount of dipoles that, in turn, are packed into the dispenser. We will use close range chaff as seduction chaff, that is, chaff that will seduce the missile read radar from the ship onto itself.

An important consideration for seduction chaff is *J/S* where *J* is the signal reflected by the radar cross section of the cloud and *S* is the radar cross section signal from a radar target within the chaff cloud or in its close proximity. In most cases *J/S* should be greater than 3 dB for successful chaff protection[20]. Now, the radar cross section for randomly orientated dipoles in regard to the direction of the electric vector of an incident wave for a particular bundle, σ, is[21]

$$\sigma = 0.18\lambda^2 N$$

where λ is the wavelength of the incident wave in metres and N is the number of effective dipoles in a bundle. However, note that the effective number of dipoles in a chaff bundle can be as few as 30% of the total dispensed. As lower limit of effective radar cross section is then

$$\begin{aligned}\sigma &= (0.18)(0.3)\lambda^2 N \\ &= 0.054\lambda^2 N.\end{aligned} \tag{3.9}$$

Assume every close range chaff bundle has 1,85 million dipoles, then it will generate a radar cross section of only 100 m^2. Furthermore, our generic FAC(M) has a worst case radar cross section of 200 m^2. This equates to 23 dBm^2. Therefore, the close range chaff's radar cross section must be 26 dBm^2 or 398 m^2 to be effective. Now, if every dispenser, that is every chaff rocket, has three bundles as described above, it will generate a radar cross section of approximately 300 m^2 which is not sufficient radar cross section for one chaff rocket to be effective against the missile head radar.

Also, for such a close range chaff rocket to be technically feasible, the dipoles must be metal-coated dielectric chaff[22]. In the event that metal-foil dipoles are used, only a fraction of the radar cross section will be generated as fewer dipoles will fit into the dispenser.

We choose close range chaff such that every rocket is capable of carrying three chaff bundles in the 0.5 cm dipole range such that every chaff bundle produces 100 m^2 of radar cross section. Furthermore, for operational reasons which will be discussed

---

[20] Van Brunt, *op. cit.* p 379.
[21] *Ibid.* p 380.
[22] *Ibid.* p 381.

later, we place the rocket launchers on either side of the superstructure such that it can fire to port and to starboard of the ship. We assume that, on command, the rockets will deploy with one second intervals and that they will dispense the chaff within a tenth of a second. Also, we assume that the burst point for the chaff is forty metres ahead of the pivoting point and, depending on which launcher was used, nine metres either to port or to starboard. When the chaff rocket dispenses the three bundles, no specific pattern will emerge, that is, the bundles may be dispensed randomly all round the burst point. However, we assume that the chaff will be displaced about two metres from the burst point. Therefore, our model will dispense the individual close range chaff bundles uniformly distributed all round the compass at two metres. The Pascal source code to achieve this follows directly.

```
procedure FireCRC;
{----------------------------------------------------------}
{ Procedure to simulate the firing of a single CRC rocket }
{----------------------------------------------------------}
var point : vectortype;
    anchor,
    angle  : real;
    index  : integer;
begin
   {------------------------}
   { Increment Chaff counter }
   {------------------------}
   ChaffNumber := ChaffNumber + 1;
   {---------------------------------}
   { Set mean relative bloom position }
   {---------------------------------}
   if CRCPort then
      begin
         angle := ship.head + arctan(CRCats/CRClong);
         if angle > 2*pi then
            begin
               angle := angle - 2*pi;
            end;
      end
   else
      begin
         angle := ship.head - arctan(CRCats/CRClong);
         if angle < 0 then
            begin
               angle := angle + 2*pi;
            end;
      end;
   point[1] := ship.posn[1] + sqrt(sqr(CRCats)+ sqr(CRClong))
               * cos(angle);
   point[2] := ship.posn[2] + sqrt(sqr(CRCats)+ sqr(CRClong))
               * sin(angle);
   {------------------------------------------------------------}
   { Place bloom positions for three bundles in Cartesian grid }
   {------------------------------------------------------------}
   for index := 1 to 3 do
      begin
         ChaffCount := ChaffCount + 1;
         anchor := radian(uniform(0,360,CRCStream));
         chafflist[ChaffCount,1] := point[1] + CRCdisp*cos(anchor);
         chafflist[ChaffCount,2] := point[2] + CRCdisp*sin(anchor);
         chafflist[ChaffCount,3] := BundleCRC;
      end
end;
```

Note that `BundleCRC` = 100.0, `CRCdisp` = 2.0, `CRCats` = 9 and `CRClong` = 40.

3.6.4 MEDIUM RANGE CHAFF

Medium range chaff (MRC) is primarily used to reduce the missile hit probability by giving it many targets to choose from. To be effective, it must be placed sufficiently far away from the ship, that is, it must be so far from the ship that the chaff and the ship will not appear in the missile range gate at any one time. Also, it should be placed towards the missile threat so as to present itself as a target worthwhile tracking to the missile before the missile detects the ship. Medium range chaff is sometimes referred to as dilution chaff.

Basically, medium range chaff is akin to close range chaff. The exception is that medium range chaff is deployed further away than close range chaff and as a result, we can argue that the rocket payload might be somewhat smaller in order to travel the longer distance. Also, medium range chaff is often deployed when a ship moves through a missile high risk area before the detection of the oncoming missile. Because of the fact that modern missiles can approach from any direction, medium range chaff is normally deployed in a pattern all round the ship. In order to keep the ship and the medium range chaff sufficiently far apart, it is deployed in the range interval [800,1200] metres. Figure 3.15 shows a typical medium range chaff pattern.

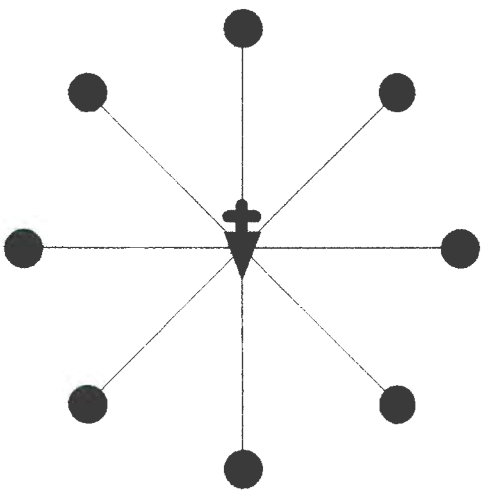


Figure 3.15: MRC pattern around a ship

For our simulation we choose a pattern of eight medium range chaff decoys as in Figure 3.15 above. We will orientate the pattern on the ship's heading, that is, we will fire one rocket on the ship's heading and the other seven rockets with 45° intervals around the compass, all at 1 kilometre. We assume that the chaff rocket's payload is three bundles of dipoles such that each bundle will produce a radar cross section of 80 m^2.

Also, we assume that the chaff rockets will not fly exactly true and that the burst point will be normally distributed in both Cartesian dimensions with a standard deviation of 25 metres. The Pascal source code for the simulation follows directly.

```
procedure FireMRC;
{------------------------------------}
{ Procedure to fire a pattern of MRC }
{------------------------------------}
var posn    : vectortype;
    bearing : real;
    index,
    n       : integer;

begin
   bearing     := ship.head;
   for index := 1 to MRCnum do
      begin
         bearing := bearing + radian(45);
         if bearing>2*pi then
            begin
               bearing := bearing - 2*pi;
            end;
         for n := 1 to MRCbundles do
            begin
               ChaffCount := ChaffCount + 1;
               posn[1] := ship.posn[1] + MRCRnge*cos(bearing);
               posn[2] := ship.posn[2] + MRCRnge*sin(bearing);
               chafflist[ChaffCount,1] := normal(posn[1], MRCSD,
                                                 MRCStream);
               chafflist[ChaffCount,2] := normal(posn[2],MRCSD,
                                                 MRCStream);
               chafflist[ChaffCount,3] := BundleMRC;
            end;
      end;
end;
```

Note that `BundleMRC` = 80.0, `MRCRnge` = 1000.0 and `MRCSD` = 1.0.

3.7 VALIDATION OF THE SHIP MODEL

In order to validate the simulation model the operations researcher should, at least, answer the following two questions in the affirmative[23]:

- Is the simulation model requisite?

- Do the outputs of the simulation model conform to the outputs of the system under consideration?

In validating a simulation model, it is necessary to at least show that the model is requisite and for known real life inputs it produces outputs that are not significantly different from the real life system's output. Phillips[24] coined the phrase "*requisite decision modelling*". He stated that a model can be considered requisite only when no new intuitions emerge from the problem or when it contains everything that is necessary for solving the problem. That is, a model is requisite when the decision maker's thoughts about the problem, beliefs regarding uncertainty and preferences are

[23] G.N. Engelbrecht, *A Guide to Numeric Simulation Techniques*. Simon's Town: SA Navy, 1996. p 279.

[24] L.D. Phillips, *"Requisite Decision Modelling"*, Journal of the Operations Research Society, 1982, 33, pp 303-312.

fully developed. In general this approach to decision modelling is endorsed in academic circles. Amongst others, Clemen[25] also advocates this view.

We scrutinised the ship model[26]. Three aspects with respect to the ship were not included in the model. The first is the ship's ability to seduce the missile's range gate by a repeater jammer or range gate stealer. Under the assumption that the MHR will employ a leading edge tracking algorithm, this equipment was considered not necessary for the simulation model. The second is the ship's ability to detect the oncoming missile by means of an electro-optical tracker. We considered the detection range of an electro-optical tracker to be considerably inferior to the detection range of the electronic support measure equipment and the search radar. Therefore, we decided not to model this equipment. Similarly, we discarded the third aspect, that is, we considered that the visual detection of the oncoming missile will take place after electronic support measure and radar detection.

We concluded that the ship model is necessary and sufficient, that is, the ship model is requisite. Furthermore, we have revisited the individual components of the model by running the simulation program with a hundred replications and we have found the simulation output to be in accordance with the real life situation as it was described in this chapter. Therefore, we have also answered the second question in the affirmative.

We conclude by saying that, because the ship model is requisite and the outputs of the simulation model conform to the outputs of the system under consideration, the ship model is assumed to be valid.

---

[25] R.T., Clemen, *Making Hard Decisions : An Introduction to Decision Analysis*, Boston: PWS-Kent, c1991, p 9.

[26] Captain J.E.G. Kamermann, SAN, Project Leader, SA Navy Corvette Acquisition Team. Personal Interview. 1 May 1997. Centurion.
Captain T.B.D. Johnson, SAN, Director Operational Test and Evaluation, SA Navy, Personal Interview. 23 May 1997. Centurion.

Chapter 4

CONCEPT MODEL OF A FIRE-AND-FORGET SURFACE-TO-SURFACE MISSILE (SSM)

4.1 PREMISE FOR SPECIFYING A GENERIC SSM

In order to specify a generic SSM, we shall investigate missiles in general to ascertain which types of missiles are most common and then specify a missile accordingly. We view SSMs from the following three perspectives:

- Homing Method.
- Propulsion.
- Flight Path.

4.1.1 HOMING METHOD

Consider the list of fire-and-forget SSMs in Appendix B. It is clear that SSMs can, according to their homing method, be categorised in four main broad categories[1], viz., radar homing, radar homing with infra-red (IR) homing as an alternative homing method, infra-red homing and a combination of radar and infra-red homing. The relevant statistics are in Figure 4.1.

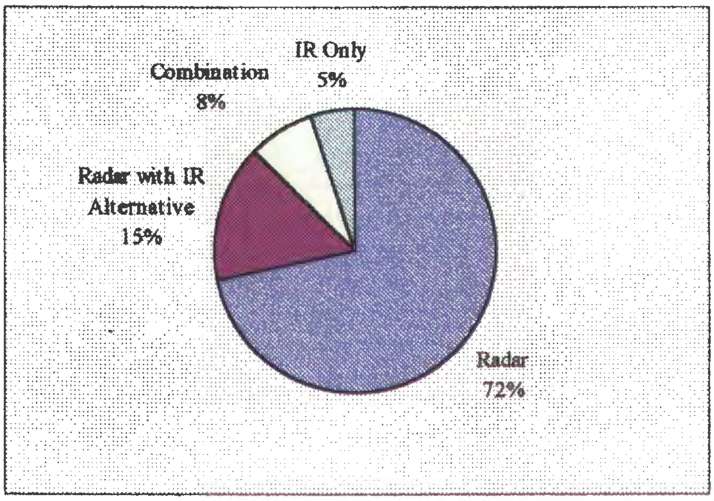


Figure 4.1: Percentage of SSM makes with associated homing methods

[1] D. Richardson, *Naval Armament*. London: Janes. 1981. pp 16-34

In all cases, these missiles are pre-programmed with the expected target position. The missile will fly to the general area where the target is expected, activate its homing device and if it acquires the target, it will proceed to prosecute same in the particular missile's unique manner.

Note that 87% of these missile types use the radar homing method as their primary or only homing method. Furthermore, it is generally accepted that the number of radar homing missiles deployed also far outnumbers other makes.

4.1.2 PROPULSION

SSMs normally have two propulsion systems. The first is called the *booster* and the other is called the *sustainer*. The booster's function is to provide the necessary power to launch the missile from the ship. It must deliver sufficient thrust to overcome the missile's inertia, lift the missile to the required height and accelerate it to the required speed. The sustainer sustains the missile's speed to enable it to fly for the duration of the engagement.

The booster is normally a solid-propellant motor whilst the sustainer can be a solid-propellant motor, turbo-jet or turbo-fan. For example, the RBS 15 missile uses a solid-propellant booster and a turbo-jet sustainer[2]. The propulsion system of the missile will not be modelled in the simulation. However, the required speed, which results from the propulsion method, is of cardinal importance.

Now, by inspection of Appendix B, we note that, of the forty three SSMs depicted there, twenty seven sustain missile speeds of between Mach 0.85 and Mach 0.9. Furthermore, six SSM sustain speeds of less than Mach 0.85, eight sustain speeds greater than Mach 0.9, whilst the sustained speed of two missiles is unknown.

4.1.3 FLIGHT PATH

In essence, there are only two ways in which a missile approaches its target. Firstly, the missile flies directly from its launch platform to the general area where the target is expected. First generation missiles used this direct approach exclusively. Secondly, with the improvement in missile motor technology, the range of missiles, in general, was extended to the extent that the new generation SSMs now have typical maximum ranges of about 135 nm. Again the RBS 15 serves as an example[3]. Therefore, it has become feasible for SSMs to make use of an indirect approach to its target. This is achieved by programming the missile to proceed through one or more way-points en-route to its target.

The tactical advantage of the latter approach is twofold. In the first place, the launch ship's position is not compromised when she fires a missile at an opponent or target. In the second place, the target can be attacked simultaneously from various directions. This leads to the saturation of the target's ability to defend itself. From a simulation

[2] Richardson. *op. cit.* p 24
[3] *Ibid.*

point of view, this implies that an oncoming missile can be expected to come from any direction.

4.1.4 GENERIC SSM

Given the discussion above, it is assumed that the representative generic SSM, which will be the object of the missile model in our simulation, has the following characteristics:

- The missile will use the radar homing method.
- The missile's cruise speed is Mach 0.9.
- The missile will have a sufficient range to allow an indirect approach on the target.

4.2 INITIAL SSM SIMULATION POSITION

Remember that when an indirect missile approach on a ship is executed, then the target ship can expect the oncoming missile from any direction. This implies that the relative bearing from the ship to the missile, b, is distributed uniformly through the compass. We therefore assume $b \sim U[0,360)$.

Now, it is reasonable to expect that the last way-point and closest point of approach (CPA) will be chosen by the opponent such that the target ship will not detect the oncoming missile until it is on its final leg. By considering the detection range of the missile by the radar in Section 3.5 and of the missile head radar (MHR) in Section 3.6.1, we estimate the range of the last way-point to be in excess of 20 nm. Note that this is a tactical consideration only. However, it is not clear at what maximum range the missile will begin its run-in on its final leg. Therefore we arbitrarily choose 28 nm as the maximum range for the missile to commence its final leg.

This leads us to choosing the commencement range of the missile's final leg as being $R_{SSM} \sim U(22,28)$. Now, by combining the generated bearing and range, the missile's initial position can be transformed into a Cartesian grid position in the normal manner.

4.3 FLIGHT PROFILE

We will discuss the following aspects in regard to the missile flight profile in this section:

- Trajectory and initial heading.
- Speed.
- Activating the missile head radar.

4.3.1 FLIGHT PATH AND INITIAL HEADING

Figure 4.2 depicts a typical plan view of the trajectory of a SSM en-route to its target. Note that there may be more than one way-point ordered by the SSM launch ship.

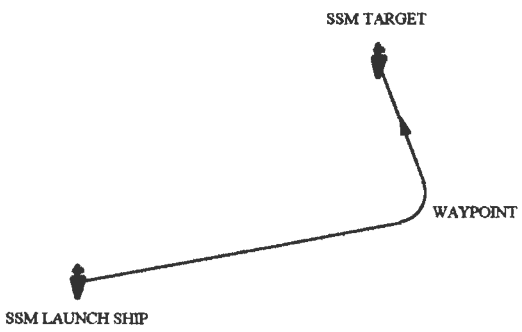


Figure 4.2: Typical plan view of SSM flight path.

Now, suppose that the missile was launched at a range of 120 nm from the SSM target ship and that is so happened that the SSM target ship altered course to a new course that was perpendicular to the SSM's final leg. Also, suppose that the missile's speed during the engagement was 600 knots and it was programmed to activate its missile head radar at 20 nm from the target position. It follows then that it would have taken the SSM ten minutes to reach the target area, that is, the position where the missile was programmed to activate its missile head radar. Furthermore, if the SSM target ship was proceeding at 30 knots, it would have displaced itself by 5 nm perpendicular to the SSM's final leg. This represents an angular movement from the missile's final leg of θ_{max}, where

$$\theta_{max} = \arctan\left(\frac{5}{20}\right),$$

or $\theta_{max} \approx 14°$ during the flight of the missile over the first 100 nm in this worst case scenario. At best, that is if the SSM target ship was on the same course to the missile's final leg or its reciprocal, we will have that $\theta_{min} \approx 0°$.

Now, by the central limit theorem[4], we assume a normal distribution for θ where $\theta_{max} \approx 14°$ is and we assume the standard deviation to be

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\theta_i - \bar{\theta}\right)^2},$$
$$\approx 1.667.$$

We use this distribution to set the SSM's heading on its final leg.

[4] S.V. Hoover and R.F. Perry., *Simulation - a Problem-Solving Approach*. Reading Ma: Addison-Wesley, 1989. p 232.

4.3.2 TRAJECTORY

From Section 2.3, the height trajectory of the missile must be such that the missile must be able to be employed for 95% of the time in the area of operations. We deduced that it means that the missile must be able to fly successfully over waves, including swell and wind waves, of up to 8.5 metres. Now, for a typical subsonic missile to operate effectively, it should fly at more than seven metres above the maximum wave heights[5]. Therefore our missile cruise height should be at least 15.5 metres above the bottom troughs of the sea surface.

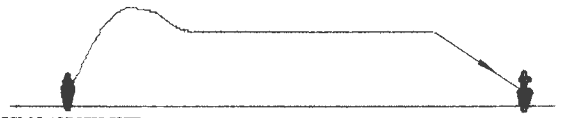


Figure 4.3: Typical side view of SSM trajectory.

The side view of a typical SSM trajectory represents three flight phases. Firstly, there is the *launch* phase. During this stage, the missile is launched with the aid of some booster motors to gain its flight speed and height. Secondly, the missile now proceeds to the target at some predetermined height. This is normally referred to as the *cruise* phase. Some missiles can fly at varying cruise heights during various times. This is normally set by the operator before launch[6]. Finally, the missile dives down on its target or dives and skims over the sea at a very low height until it impacts with its target. This is known as the *terminal* phase.

For the purposes of our simulation, we assume that during the cruise phase, the missile will cruise at a height of 20 metres above the sea. Also, we assume that the missile will dive down on its target to impact at zero height at zero range. However, we will not simulate the height of the missile as we assume that it is irrelevant to the outcome of the simulation. We can make that assumption because we will choose our hit/miss criteria such that it will take the non-simulation of the terminal phase trajectory into account.

4.3.2 SPEED

The *Mach-number* is a quantity without any dimension that relates the speed at which sound is propagated through air, a m/s, to the real air speed of an object, V m/s, through the same medium. This relationship[7] can be expressed as

---

[5] Gene M. Jordan, Chairman of FAAC Inc., Ann Arbor, USA. Personal Interview. September 1996. Singapore.

[6] Åke Svensson., Project Manager, Missile Division, Saab Dynamics AB, Linköping, Sweden. Personal Interview, 22 April 1997. Pretoria.

[7] M.J. Zucrow, and J.D. Hoffman, J.D., *Gas Dynamics Volume 1*. New York: John Wiley, 1976. pp 63 - 66

$$M = \frac{V}{a}, \tag{4.1}$$

where M is the Mach-number. Furthermore, the speed of sound in air can be obtained by the equality

$$a = \sqrt{\gamma R t} \tag{4.2}$$

where γ is a constant representing the ratio between specific heat capacity at constant pressure and volume respectively with $\gamma = 1.4$, the universal gas constant $R = 287.06$ J/kgK and t is the static air temperature in degrees Kelvin.

The universal gas law controls the ratio between pressure, density and temperature. However, the influence of changes in barometric pressure on air density is very small and therefore it may be omitted as a factor[8]. Also, the effect of temperature in the range 270°K to 300°K on γ is omissible because of its extremely small effect on same[9]. Therefore, we can assume $\gamma = 1.4$ in our simulation.

Thus, in our simulation, a can be considered dependent on temperature only and we use the model

$$\begin{aligned} V &= Ma \\ &= M\sqrt{\gamma R t} \ \text{m/s} \end{aligned} \tag{4.3}$$

to simulate the speed of the missile.

By inspection of Table B-2, we see that the majority of missiles fly at subsonic speeds. Also, we note that the majority of subsonic missiles fly at Mach 0.9. Therefore, we choose Mach 0.9 as our generic missile speed.

4.3.3 ACTIVATING THE MISSILE HEAD RADAR

In order to decide on when to activate the missile head radar, two factors are important, viz., the range at which the missile head radar will detect a target and the reaction time for the target to carry out some tactic against the missile. To investigate the detection range of the missile head radar, we must first specify the radar. The modern tendency is for missile head radars to operate in as high a frequency band as technology will allow. For example, the new RBS 15 Mk 3 missile operates in the Ku radar band[10]. Also, because the missile has a limited battery capacity, the missile head radar will invariably not have a very high peak power output. We estimate a representative peak power output to be in the order of 20 kW[11].

---

[8] Zucrow and Hoffman. *op. cit.* pp 700 - 701.
[9] *Ibid.*
[10] Åke Svensson, *op. cit.*
[11] *Ibid.*

We choose the following missile head radar parameters:

The simulation uses the following constants:

Radar Type	: Tracking Radar.
Transmitter	
Wavelength	: 1 cm.
Peak Power	: 20 kW.
Pulse Length	: 1 μs.
PRF	: 500 hz.
Tx Loss	: 1 dB.
Receiver	
Noise Level	: 7 dB.
System Losses	: 9 dB.
Rx Losses	: 6 dB.
Pulses Integrated	: 10.
Integration	: Coherent.
Antenna	
Gain	: 28 dB.
Vertical Beam Width	: 7°.
Horizontal Beam Width	: 1.1°.
Side Lobe Level	: -15 dB.
Height	: 20 m.
Polarisation	: Vertical.
Discrimination	
Type	: Signal Processing.
Processing Losses	: 2 dB.
Errors	
Imp Height Error	: 3 m.
Timing Jitter	: 9 ns.

If we consider the missile head radar detection ranges obtained by the Justice Radar Simulation and reported in Appendix D, we see that the missile head radar will not detect our generic FAC(M) targets outside 29.7625 kilometres (See Table 4.1). Also, with the exception of sea states 5 to 7, the generic FAC(M) will be detected with a probability of 100% by the time the missile is 13.1875 kilometres from its target.

We note that the optimum probability of detection, P_{detect}, is somewhat reduced for sea states 5, 6 and 7. Although this will influence the missile's performance adversely, we will only consider $P_{detect} = 1$ as we are not interested in missiles that do not detect our FAC(M). Now, from Table 4.1, the optimum range at which $P_{detect} = 1$ is consistent, is 13.1875 kilometres. For the purposes of our simulation, we assume that it is the case for all sea states. Therefore, we conclude that the generic SSM will detect the generic FAC(M) at 13.1875 kilometres with probability 1.

Now, in designing the missile, the designer must consider the target's tactics and the reaction time that the target will have from detecting the missile head radar until missile impact. The objective is to minimise the target's reaction time. This statement implies that the missile head radar must be activated as late as possible.

Sea State	Optimum P_{Detect}	Optimum Range (km)	Maxiumum Range (km)
0	1.00	13.1875	29.7625
1	1.00	13.1875	29.7625
2	1.00	13.1875	29.7625
3	1.00	13.1875	29.7625
4	1.00	13.1875	29.7625
5	0.97	15.1375	29.7625
6	0.95	17.0925	29.7625
7	0.94	18.3648	29.7625

Table 4.1: Justice Radar Simulation results for generic MHR against generic FAC(M)

Recall from Section 4.3.1 that the ship can be displaced as much as five nm or 9 216 metres from its original position. In order to maximise the missiles detection probability in the search phase and to minimise the target reaction time, the best the missile designer can do, is to add the ship's displacement during the missile's flight and the optimum detection range of the missile head radar. Therefore, to satisfy both constraints the missile head radar should activate at a range of 22 403.5 metres. For our simulation, we choose the missile head radar activation range as 22 kilometres from the FAC(M)'s original position which is the position (0,0).

4.4 THE MISSILE HEAD RADAR RANGE GATE

Tracking radar resolves its tracking problem by producing a gate pulse such that the receiver will only open for a very specific time period. This produces a gate in range that the tracking radar will investigate for radar reflected energy. Figure 4.4 shows the principle. Note that the convention followed shows time in reverse order so as to accommodate the showing the leading edge of the range gate generation pulse in a leading position.

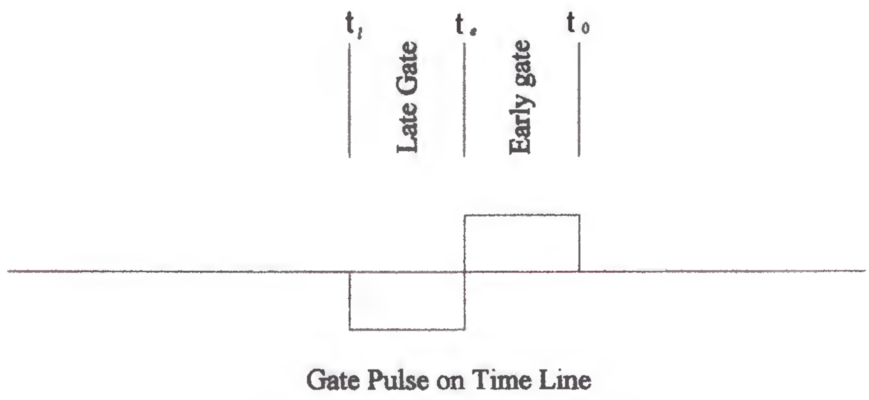


Figure 4.4: Range gate generation[12]

[12] M.I. Skolnik, *Introduction to Radar Systems*, 2 ed., Auckland: McGraw-Hill, 1981. p 177.

4.4.1 INTEGRATING THE RANGE GATE

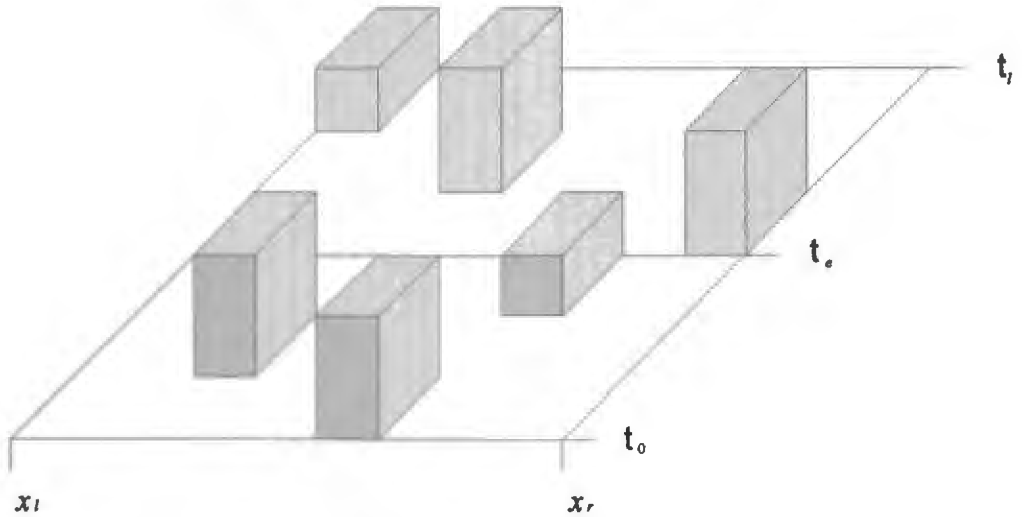


Figure 4.5: Three-dimensional view of the range gate

The range gate width depends on the radar beam width. Normally, the width of the range gate is limited to the −3 dB points of the main radar lobe. As the −3 dB points are almost parallel at extended ranges, we will assume that, for the purposes of our simulation, it is indeed so. Furthermore, we will denote the left hand edge of the range gate as x_l and the right hand side of the range gate as x_r. Now, Figure 4.5 is a graphical representation of the range gate. The shaded boxes indicate radar reflective energy originating from in the range gate.

By the *integration* of the range gate, we mean establishing how much energy is reflected from the area covered by the range gate which is greater than the noise floor of our radar. The amount of energy in the early gate, E_e, is given by the volume of the boxes representing the reflected radar energy from the early gate, or

$$E_e = \int_{x_l}^{x_r} \int_{t_0}^{t_e} f(x) f(t) dt dx .$$

Similarly, the amount of energy in the late gate, E_l, is given by the volume of the boxes representing the reflected radar energy from the late gate, or

$$E_l = \int_{x_l}^{x_r} \int_{t_e}^{t_l} f(x) f(t) dt dx .$$

However, in Section 3.3, we have represented the radar cross section of our FAC(M) as 17 points in two-space, each with its own radar cross section. In order to simulate the integration of the range gate, we simply substitute the integrals with a summation such that

$$E_e = \sum_{i=1}^{n} e_i$$

where the e_i refers to energy received from the ith radar reflective point in the early gate and

$$E_l = \sum_{j=1}^{m} e_j$$

where the e_j refers to energy received from the jth radar reflective point in the late gate. Now, in the case for the angular errors, the method to find the energy emanating from the left and right gates respectively is similar.

4.4.2 FINDING THE RANGE TRACKING ERROR

A tracking radar cannot differentiate between the distances of particular sources of reflected energy from t_e; it merely establishes that a particular source of reflected radar energy is in a particular gate. Therefore, if the range gate is positioned precisely over a target the expected values of E_e and E_l must be of the same magnitude. Now, consider Figure 4.6. If $E_e > E_l$, then it is assumed that t_e was placed too far away from the radar antenna.

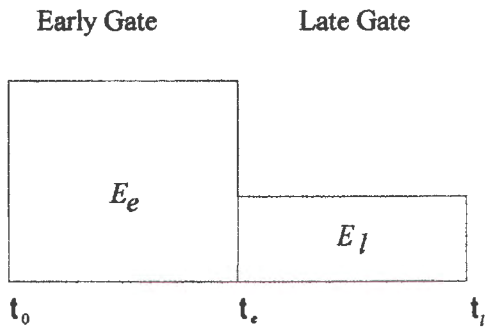


Figure 4.6: Graphical representation of the range dimension of the range gate

The question that must now be answered is: How far must the range gate move towards the radar antenna or what is the magnitude of the range error? Note that if $E_e > E_l$, then the range error, r_e, must be negative and when $E_e < E_l$, then r_e must be positive. This convention is followed so that the error can be applied to the present range without testing for the direction of the error. By simple algebraic manipulation we find that

$$r_e = \frac{(E_l - E_e)}{2(E_l + E_e)}\left(\frac{t_l - t_0}{2}\right). \tag{4.4}$$

This equation will hold for as long as the early and the late gate are of equal size. If they are not and the early gate is smaller than the late gate, as we will see when we discuss leading edge tracking, then we need to make modifications to (4.4).

Firstly, we must ensure that the relative weight of the potential radar energy in both gates is equal. This is achieved by setting

$$E'_e = \left|\frac{t_l - t_e}{t_e - t_0}\right| E_e \tag{4.5}$$

and

$$E'_l = E_l . \tag{4.6}$$

Secondly, we modify (4.4) to read

$$r_e = \frac{(E'_l - E'_e)}{2(E'_l + E'_e)}(t_e - t_0) \tag{4.7}$$

in the case where the most energy is in the early gate, and

$$r_e = \frac{(E'_l - E'_e)}{2(E'_l + E'_e)}(t_l - t_e) \tag{4.8}$$

in the case where the most energy is in the late gate.

However, amplifying the energy levels in the two gates linearly will result in excessive range gate jitter, that is, the range gate will oscillate in range more violently than what is desired by the SSM tracking modules. To overcome this problem, logarithmic amplification can be used. In our simulation, we could do this by setting

$$E''_e = 10\log_{10}(E'_e) \tag{4.9}$$

and

$$E''_l = 10\log_{10}(E'_l), \tag{4.10}$$

that is, we convert the summed radar cross section values in the two gates from m^2 to dBm^2 when the summed radar cross section value is equal to or larger than 1.

Finally, in our simulation, we deviate from (4.9) and (4.10) and use the following equality to implement a logarithmic amplifier[13]

$$\begin{aligned} y &= 10(\ln(x) + \log_{10}(e)) \\ &= 10(\ln(x) + 0.434294482) \end{aligned} \tag{4.11}$$

[13] Dr. T de Wet, OR practitioner, Institute for Maritime Technology, Simon's Town, Personal Interview, May 1997.

where y is the output from the logarithmic amplifier for $x > 1$. Otherwise, $y = 0$.

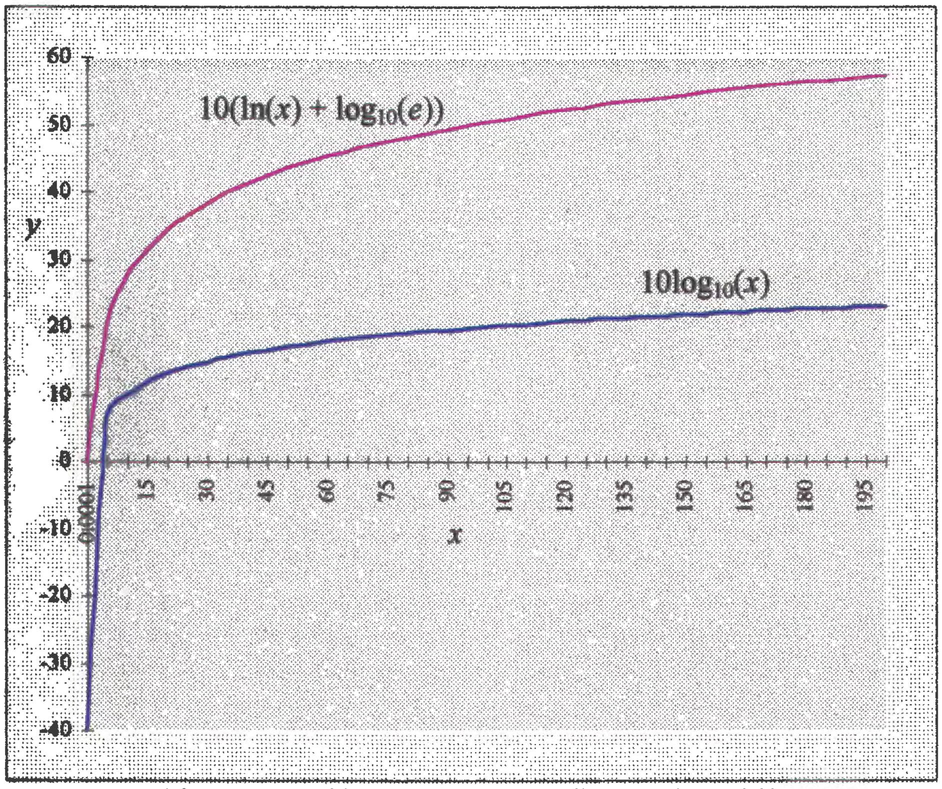


Figure 4.7: $10\log_{10}(x)$ versus $10(\ln(x) + \log_{10}(e))$

Note, from Figure 4.7, that the graphs of (4.10) and (4.11) are both logarithmic in nature. However, the graph of (4.11) initially grows faster than the graph of (4.10) whilst the slope of the (4.11) is larger than the slope of (4.10) over the interval [1,200]. Therefore, it can be expected that the resultant tracking error will result in less range gate jitter than when linear amplification is used, but that the tracking errors would be more responsive than when (4.9) and (4.10) are used.

4.4.3 FINDING THE LINE TRACKING ERROR

To find the error in line of the range gate in regard to the target, we proceed as previously for the range tracking error. However, by estimating the angular movement rate from the situation in the range gate, we can smooth the angular tracking somewhat. In this simulation, if there is observed radar energy in either the left or right gates only, we estimate the angular tracking rate between pulses to be one milli-radian in the direction of the half gate which contains the radar energy. Furthermore, if there is observed radar energy in both the left or right gates only, we estimate the angular tracking rate between pulses to be 0.5 milli-radian in the direction of the half gate which contains the most radar energy.

4.4.4 RANGE GATE IMPLEMENTATION

The Pascal source code to implement the resolution function of the range gate is in Appendix G. The relevant function and procedures are as follows:

Procedure/Function	Name	Pages
`function`	`logamp`	G-7
`procedure`	`CalculateRangeError`	G-14
`procedure`	`CalculateAngleError`	G-15
`procedure`	`transform`	G-15
`procedure`	`ResolveRangeGate`	G-16

Note that, in order to resolve the range gate in our simulation, the positions of the radar cross section reflective areas must be transformed from our Cartesian grid to a grid with the range gate centre position and the gate direction as reference.

4.5 SEARCH PHASE

On activating the missile head radar, the missile must search for its target. Because of its physical nature, the missile head radar antenna rotation is normally restricted. We assume that our generic SSM missile head radar antenna is restricted to rotate 25° on either side of the missile's nose direction. This means that the missile is restricted to search in the arc which lies within 25° of the missile's heading. Also, we choose the search to start at half the maximum rotatable angle, that is 12 ½°, to the left of the missile's heading (See Appendix H for further remarks).

From the Justice Radar Simulation we deduced that optimum range at which $P_{detect} = 1$ is consistent, is 13.1875 kilometres. This implies that the missile's maximum search distance should be less than 13.1875 kilometres. We choose the generic missile's maximum search distance to be 12 kilometres. Furthermore, we arbitrarily choose the distance from the missile head radar where the search will start to be 6 kilometres.

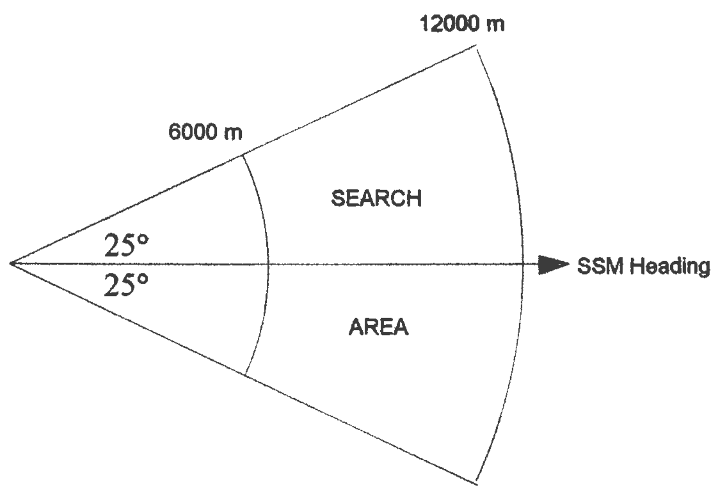


Figure 4.8 : SSM search area.

We set the range gate according to the above criteria by the following Pascal code:

```
Procedure SetSearchGate (angle,value : real);
{--------------------------------------------------------------------}
{ Procedure to set up the initial position of the MHR range gate }
{--------------------------------------------------------------------}
begin
   with SSM.gate do
      begin
         direction := SSM.head + radian(angle/2);
         if direction > 2*pi then
            begin
               direction := direction - 2*pi;
            end;
         range     := value;
         GateWidth := range * tan(radian(Halfbeam));
         right     := TRUE;
         up        := TRUE;
      end;
end;
```

The range gate is now set to search to the right and away from the missile. We chose the horizontal beam width of the missile head radar, in Section 4.3.3, to be 1.1°. Therefore, we can step the missile left or right at 1° per pulse and still maintain an overlap between the searched areas or searched blocks. Thus, one sweep of the antenna from the left to the right limit, or vice versa, would take 50 pulses or $\frac{1}{10}$ seconds.

Although this is somewhat faster than what a mechanical device would allow, if we assume a fixed missile head radar antenna with electronic beam steering[14], this would be possible. Also, tracking radar would normally illuminate every search block several times to enhance the radar's detection probability of targets in the search block, that is, to allow for the statistical nature of the radar equation[15].

In our case, it is submitted that it is not necessary to simulate several pulses in a particular search block as we assume the radar reflective energy from the target to be constant at its expected value and thus that the radar will see the target on every pulse. (See Section 3.3)

Furthermore, we choose an increase in range gate range of 2 metres between pulses. This will allow for the search to continue up to the missile head radar's maximum range. The missile head radar will reach its maximum range after 3 000 pulses or 6 seconds. Thereafter, it will reset the range gate range to 6 kilometres and commence searching out again. Note that although the range gate moves, relative to the missile, six kilometres in range over six seconds, the range gate moves, relative to the environment, 7 860 metres in the six seconds if we assume the missile's velocity to be 310 m/s. The Pascal source code to simulate a pulse in the search mode, has been named `procedure SendSearchPulse` and is in Appendix G at pages G-19 to G-21.

---

[14] A. Farina and F.A. Struder, *Radar Data Processing - Volume 1 - Introduction and Tracking*. Letchworth: Research Studies Press,1985, p 19.

[15] *Naval Operations Analysis*. 2 ed. Annapolis, Ma: Naval Institute Press. 1977. pp 86-93

Note that the range gate width is a function of the range gate range and the beam width. The test value `GateWidth` is the value in metres of half the gate width. Furthermore, once the missile head radar finds sufficient radar reflective energy in its range gate, that is, there is radar reflective energy in either the late gate or in the early gate or in both gates, the missile logic is changed to the acquisition phase whilst the range gate is adjusted towards the target at the same time.

4.6 ACQUISITION PHASE

In order to acquire the target, that is, to solve the tracking problem to such a degree that target can be smoothly tracked and the missile guided onto the target, two aspects are important, viz.,

- the early and late range gates must be sufficiently small so that the consecutive range adjustments are also as small as possible; and

- the rate of range ($\dot{r}$) and angular ($\dot{\lambda}$) movement of the range gate in regard to time must be calculated.

In order to address the first aspect, when the missile commences the acquisition phase, the range gate is reduced by multiplying the present range gate by a factor (`AcqFactor`) of 0.55 for every consecutive pulse until the gates are both 40 m in depth. The algorithm for this follows directly in Pascal source code.

```
AcqGate   := AcqGate * AcqFactor;
if AcqGate < 40.0 then
   begin
      AcqGate := 40.0;
   end;
```

Once the early and late range gates are 40 metres in depth, the second aspect can be addressed. The range error (r) and the angular estimate (λ) is pushed into a stack at every resolution of the range gate. Once the stack with stack size (n) is full, $\dot{r}$ and $\dot{\lambda}$ are calculated by

$$\bar{r} = \left| \frac{1}{n} \sum_{i=1}^{n} r_i \right| \text{ m}$$

and

$$\dot{r} = \bar{r} f \text{ m/s}$$

where f is the pulse repetition frequency,

$$\bar{\lambda} = \frac{1}{n} \sum_{j=1}^{n} \lambda_j \text{ radians}$$

and

$$\dot{\lambda} = \bar{\lambda} f \text{ radians/s.}$$

In the event that no radar reflective energy is found in the range gate for any particular pulse whilst the missile is in the acquisition phase, the acquisition of the target will be aborted. The missile will return to the search phase whilst setting the range gate range at the last known target range.

4.7 TRACKING PHASE

During the tracking phase, the missile head radar will continue to track the target with both the early and the late gates set to a depth of 40 metres. The half-width of the range gate (w_r) will remain

$$w_r = d \tan\frac{\theta}{2}$$

where d is the range of the centre of the range gate and θ is the horizontal beam width of the missile head radar. Once the range of the missile head radar's range gate is less than 2 000 metres, we assume that the missile head radar will also get target reflections from the main side lobes[16]. The half-width of the range gate is now adjusted to

$$w_r = 3d \tan\frac{\theta}{2}$$

to allow for one full side lobe. This will allow for tracking a larger section of the target at closer ranges. Experimentation with the simulation model showed that it is sufficient to model the main side lobes only. Therefore secondary side lobes were not considered.

In the tracking phase, the rates $\dot{r}$ and $\dot{\lambda}$ are be calculated after every n pulses. The use of $\dot{r}$ and $\dot{\lambda}$ will be explained in Section 4.9 Again, if no radar reflective energy is found in the range gate for any particular pulse whilst the missile is in the tracking phase, the tracking of the target will be aborted and the missile will return to the search phase whilst setting the range gate range at the last known target range. The two stacks are also set to empty to facilitate the next acquisition phase.

Once the missile head radar's range gate range is less than 150 metres, the missile stops tracking the target and commences the ballistic phase. This phase will be dealt with in Section 4.8.

As the acquisition and tracking phases are very similar in nature, both were accommodated in a single procedure for simulation purposes. The Pascal source code to simulate a pulse in the acquisition and tracking phases, has been named `procedure SendTrackPulse` and is in Appendix G at pages G-21 to G-23.

---

[16] Skolnik., *op. cit.* pp 160-167.

4.8 BALLISTIC PHASE

During the last 150 metres of the missile flight path, the missile is in the ballistic phase. In the simulation, this is achieved by the Pascal source code that follows directly.

```
{---------------------------------------------------}
{ Assume ballistic flight path for final 150 metres }
{---------------------------------------------------}
if NOT(BALLISTIC) and (SSM.gate.range<150.0) then
   begin
      BALLISTIC := TRUE;
   end;
```

The implication is that the missile will discontinue tracking the target to avoid violent alterations of the range gate direction and the resultant violent alteration of the missile's course in the last 150 metres. Therefore, the missile will continue to fly in the last ordered direction at a low level so as to collide with the target. In our simulation, this command is irreversible and should the missile not hit a target, it will continue to fly in the ordered direction until it runs out of fuel or propellant.

To avoid this logic in the initial stages of the simulation and the resultant rogue behaviour by the missile, the missile's range gate range is set to some value that is in excess of 150 metres when the missile is en-route to the target and before the search logic is imposed on it.

4.9 MISSILE GUIDANCE

Homing guidance is a general term used to describe guidance systems in which the target location relative to the missile is sensed by equipment on board the missile and this information is then used to steer the missile to a collision with the target, that is, the missile homes on the target. If the target is stationary, this can be accomplished by continuously pointing the missile velocity vector at the target. However, this implies that the target position must be continuously measured. Now, if the target is moving, in order for the missile to collide with the target, the missile velocity vector will continuously change. This form of homing guidance is called *pursuit guidance*.

Jordan[17] states that a missile in pursuit of a moving target will invariably manoeuvre violently during the final phases of the engagement. Therefore, our generic missile will pursue the target only in the acquisition phase. During this time, the rates $\dot{r}$ and $\dot{\lambda}$ are still not known. However, in order to close the target more directly, the present direction of the range gate centre becomes the ordered flight direction of the missile.

Now, the imaginary line in two-dimensional space on the surface of the earth from the missile to the target is called the *line-of-sight* (LOS). If we assume that both the target and the missile proceed at constant speeds and that the target will not alter course, the

[17] G.M. Jordan, *Class Notes for a Short Course in Homing Guided Missiles*, Ann Arbor, Mi: FAAC. c1975, p 2.

collision heading for the missile is found by solving the classic velocity triangle (See Figure 4.9).

It follows that when the missile is on a collision course with the target, the line-of-sight will remain constant, that is, the line-of-sight will not rotate in two-space but will only translate with the passing of time (See Figure 4.9b).

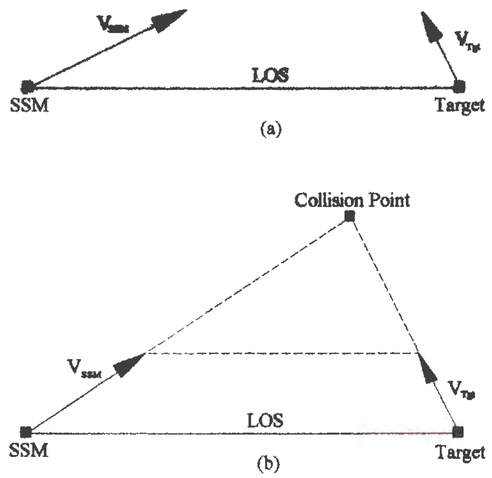


Figure 4.9 : Collision course geometry

A better method, known as the *classic proportional navigation homing* method is often employed in modern missile technology. An approach that will make proportional navigation homing possible[18], involves tracking the line-of-sight, measuring the line-of-sight rate and making the missile fly such that the line-of-sight rate is zero in order to collide with the target. Now, if the line-of-sight rate is not zero, the missile velocity vector must be turned in such a way as to reduce the line-of-sight rate. This can be achieved by turning the velocity vector at a rate proportional to the line-of-sight rate, $\dot{c}_{SSM}$, that is,

$$\dot{c}_{SSM} = \Gamma \dot{\lambda} \tag{4.12}$$

where Γ is a proportional navigation constant and $\dot{\lambda}$ is the line-of-sight rate of the target relative to the missile[19]. If we consider the time period over which the velocity vector will turn at a rate proportional to the line-of-sight, Δt, we can modify (4.12) such that

$$c_{SSM} = \Gamma \dot{\lambda} \Delta t \tag{4.13}$$

where c_{SSM} is the ordered flight path offset from the line-of-sight or *lead angle*.

[18] G.M. Jordan, *op. cit.* p 4.
[19] *Ibid.* p 4.

We note that, in the tracking phase, the missile has already calculated $\dot{\lambda}$ and $\dot{r}$. Also, the range to the centre position of the range gate is known. Furthermore, we define the time increment

$$\Delta t = \frac{R_g}{\dot{r}} \tag{4.14}$$

where R_g is the range to the centre position of the range gate and $\dot{r}$ is the range rate or relative closing speed of the target. Thus Δt is the time of flight remaining for the missile to reach the collision point with the target.

If we can find Γ such that it will stabilise the line-of-sight, we have solved the proportional navigation problem.

We now redefine (4.13) as

$$c_{\text{SSM}} = \Gamma \frac{R_g}{\dot{r}} \dot{\lambda}. \tag{4.15}$$

By experimenting with the simulation model we find that for $\Gamma = 0.12$, we have stabilised the line-of-sight to a degree where for one hundred replications of the simulation and where the ship continued on its course and speed, the mean miss distance, that is the distance from the centre of the ship to the closest point of approach by the missile, was 1.775 metres with a standard deviation of 1.611. Thus, a 95% confidence level for the mean miss distance is [1.459,2.091] metres. Furthermore, the minimum miss distance was 0.07 metres whilst the maximum miss distance was 6.6 metres.

Therefore, we accept

$$c_{\text{SSM}} = 0.12 \frac{R_g}{\dot{r}} \dot{\lambda} \tag{4.15}$$

as sufficient to implement proportional navigation by setting the ordered course for the missile to

$$C_o = C_p + c_{\text{SSM}} \tag{4.16}$$

where C_o is the new ordered missile course and C_p is the previously ordered missile course. In our simulation, c_{SSM} is referred to as the *lead angle*. By this is meant the angle, in radians, by which the missile must lead the line-of-sight in order to collide with the target.

4.9.1 MISSILE AERODYNAMICS

Missile aerodynamics is normally classified. In order to simulate the missile altering course, the rate at which the missile will turn was chosen as 70° per second. This turning rate is based on the turning characteristics of the SKUA unmanned high speed drone[20]. The drone is capable of speeds in excess of Mach 0.85 and its turning characteristics would suggest that a turning rate of 70° per second for our generic missile is acceptable. Note that this can be considered as a simulation of a first order feedback control system.

By running the simulation with 100 replications and choosing an alter course rate of 35° per second for our generic missile, we found the mean miss distance to be 1.805 metres with a standard deviation of 1.61 metres. A 95% confidence interval for the miss distance is then [1.489,2.121].

If we compare this with the previous results where the missile's turning rate was 70° per second, we see that the two confidence intervals overlap and we deduce that, at the 5% level, there is no significant difference between the results of the simulation with missile turning rates of 70° and 35° per second respectively.

```
MON 16/06/97 12:29:02

SYSTAT VERSION 5.0
COPYRIGHT, 1990-1994
SYSTAT, INC.
MON 16/06/97 12:43:03   C:\SYSTATW5\SSMTURN.SYS

 PEARSON CORRELATION MATRIX

                         DEG70       DEG35

      DEG70              1.000
      DEG35              0.991       1.000

 NUMBER OF OBSERVATIONS:  100
```

Figure 4.10: Correlation between miss distances for missile turning rates of 70° and 35° per second respectively

Furthermore, from Figure 4.10, we see that there is a strong positive correlation between the respective replications of the two simulation runs with missile turning rates of 70° and 35° per second respectively. We conclude by assuming that the model is insensitive for missile turning rates between 70° and 35° per second respectively. Therefore, the chosen missile turning rate of 70° per second is considered valid.

[20] Brig. G. Havenga., Product Manager, Kentron Ltd., Personal Interview, 2 June 1997, Centurion.

4.10 HOME-ON-JAM

The missile operation is dependent on the fact that it must be able to measure the radar reflective energy from the target in the range gate to be able to track the target. Now, suppose that the target ship jams the oncoming missile and that the missile is outside the jammer's self-screening range. This will result in the missile head radar observing large and equal amounts of radar energy in both early and late gates. This would produce zero range errors in the missile head radar. Thus, the missile would be unable to track the target in range.

In regard to the left and right gates, the situation is somewhat different. If the jammer's signal is of equal strength in both the left and right gates, this would result in a zero error in regard to line being generated. Furthermore, this would reflect the true situation because it will mean that the missile head radar antenna is pointing directly at the jammer. However, if the missile head radar antenna is not pointing directly at the jammer, then the amount of radar energy in the left and right gates will differ. For example, if the antenna is pointing to a position to the left of the jammer, one can expect that there will be a stronger jamming signal in the right gate. The result would be a right error being generated which, in turn, will result in the missile head radar antenna being trained right until the antenna points at the jammer. This fact allows the missile head radar to carry out an electronic counter-counter measure (ECCM) known as home-on-jam (HOJ)[21].

4.10.1 HOME-ON-JAM IN THE SEARCH PHASE

When the missile head radar is in the search phase, the range gate is searching up from 6 000 to 12 000 metres, resetting to 6 000 metres when it reaches 12 000 metres. In order to deal with the range problem, when home-on-jam is selected, the range gate will remain at its last known position whilst the missile head radar will track the target in angle only. The missile will now be guided by the pursuit mode. When the jamming ceases, and there is radar reflective energy in the range gate, the target will be acquired. If there is no radar reflective energy in the range gate when the jamming ceases, the missile will commence the search phase with the range gate at its last ordered range.

However, suppose that the ship ceases to jam the missile head radar when the missile's range from the ship is less than 6 000 metres. In this case, the missile will commence a search beyond the target. If we assume that when the missile is being jammed it will generate a dive command at some earlier time which is dependent on the pre-launch information that the missile has received then the missile, although it is not tracking the target, might still collide with it.

Suppose the ship has maintained her course and speed. In Section 3.4.1 we assumed the ship's course to be uniformly distributed in the interval [0,360) degrees and that the ship's speed is uniformly distributed in the interval [22,28] knots. The expected value of the ship's speed is therefore 25 knots or 12.8 m/s. Furthermore, we assume that the missile course at the time that jamming ceases is the bearing of the jammer

[21] L. B., Van Brunt, *Applied ECM - Vol 1*. Dunn Loring, Va.: EW Engineering, c1978, pp 373-376.

which is situated more or less in the centre of the ship. Also, we have specified our generic missile logic such that it will remain on the last ordered course whilst it is in the search phase. Given the above, the probability that the missile will collide with the ship is given in Figure 4.11.

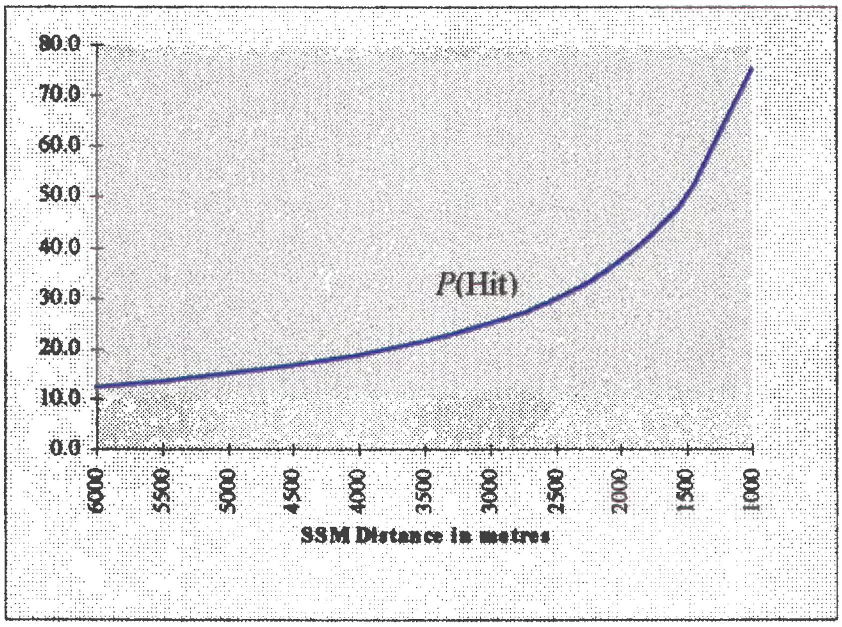


Figure 4.11: SSM probability of collision with target ship and when jamming ceases at missile-target range of less than 6 000 metres

Note that in our simulation, we will disregard this situation.

4.10.2 HOME-ON-JAM IN THE ACQUISITION PHASE

If the missile head radar is in the acquisition phase, the range gate range will correspond with the target range when the jamming commences. However, the range rate, $\dot{r}$, will not be known. Again, in order to deal with the range problem, when home-on-jam is selected, the range gate will remain at its last known position whilst the missile head radar will track the target in angle only whilst the missile is guided by the pursuit mode. Once the jamming ceases and there remains radar reflective energy in the range gate, the missile will continue to acquire the target. However, if no radar reflective energy is present in the range gate at that time, the missile head radar will return to the search mode.

Note that, because the missile is in the pursuit phase when the missile head radar's logic is home-on-jam, the missile is guided onto the target in the pursuit mode and subsequent missile course adjustments, once jamming has ceased, are expected to be small.

4.10.3 HOME-ON-JAM IN THE TRACKING PHASE

If the missile head radar is in the tracking phase when jamming occurs, then it has already calculated $\dot{r}$. Thus, the missile tracks the target in angles and adjusts the range gate range according to the last calculated value of $\dot{r}$. This will enhance the search and acquisition time of the missile head radar when the jamming ceases. In this case, the missile, once again, goes into the pursuit phase.

The simulation home-on-jam logic will be dealt with in Chapter 6.

4.11 HIT CRITERIA

We assume that the missile's warhead is triggered by three separate means, viz.,

- direct action;
- deceleration; and
- proximity fuse.

Furthermore, we assume that the proximity fuse will be enabled when the missile does not directly hit the target or the missile is not decelerated by the target. In the case of our simulation, the half length of the FAC(M) is 30 metres. For the direct action fuse to activate the warhead, it must hit the target directly. For the deceleration fuse, it must graze the target. If we assume that a near miss of one metre would mean that the missile grazes the target, then the effective half length of the target is 31 metres. Furthermore, we assume that the proximity fuse would activate the warhead if the missile arrives at its calculated point of impact plus some built in delay time and it does not impact or grazes the target and it comes within 31 metres of the centre of the FAC(M).

To discriminate between a hit and a miss, by simple geometry, we accept that, if the target range opens, the missile is moving away from the target. Thus, we test whether the range is opening or closing after every pulse. If the range is opening and the missile is within 31 metres of the FAC(M)'s centre position, we assume a hit. If the range is opening and the missile is outside of 31 metres from the FAC(M)'s centre position, we assume that the missile has missed the target. The distance at which the missile is from the FAC(M) centre position at that time, is the miss distance. Therefore, a miss distance of less than 31 metres constitutes a hit and a miss distance in excess of 31 metres constitutes a miss.

4.12 VALIDATION OF THE MISSILE MODEL

Three aspects are worthy of mentioning. Firstly, as far as the integration of the radar reflective energy in the range gate is concerned, the model can be considered adequate. An aspect that did not receive any attention, is the fact that the pulse length of the missile head radar influences the amount of energy in the respective gate halves. However, as the chosen pulse length is of a very short duration, that is, the pulse length is only 1 μs, the method of summing the radar reflective values for all points in the applicable gate was considered adequate.

Secondly, the fact that a leading edge tracking algorithm was not developed was also considered. The radar cross section model of the FAC(M) is relatively sparse in the sense that only seventeen radar cross section points were considered. Whilst experimenting with such an algorithm, the very narrow early gate normally used by missile head radars to implement leading edge tracking proved to be problematic in that, because of the sparse radar cross section FAC(M) model, the amount of radar energy observed by the model in early gate were often zero when it should have contained some radar energy. In turn, that gave rise to range gate errors in the wrong direction with the resultant degrading in the smooth tracking of the target. As we have not modelled the effects of a range gate stealer on board the FAC(M), the leading edge tracking algorithm was considered unnecessary.

Thirdly, the aerodynamic model was considered. Normally, the turning characteristics of a missile could be more accurately represented by a linear constant second order differential equation , that is, a differential equation of the form

$$\frac{d^2y}{dt^2} + 2\zeta\omega_n\frac{dy}{dt} + \omega_n^2 y = \omega_n^2 x$$

where the constant ζ is the damping ratio and ω_n is the undamped natural frequency of the missiles turning characteristics. Two aspects were considered to ascertain whether such an approach would ensure a superior result. On the one hand, without access to some classified data, ζ and ω_n would be virtually impossible to estimate. On the other hand, by simulating the missile's turning characteristics simply by letting the missile alter course at a predetermined rate, is relatively easy to implement. Furthermore, the accuracy of the simulation, by choosing a constant second order differential equation with doubtful parameters, was considered inferior to simulating a course alteration by a constant turning rate.

After careful consideration, we[22] came to the conclusion that our thoughts about the missile model as a system at the appropriate level, can be considered fully developed. Thus, the missile model is regarded as requisite. Hence, we assume the missile model to be valid.

[22] Captain T.B.D. Johnson, SAN, Director Operational Test and Evaluation, SA Navy, Personal Interview. 18 June 1997. Fish Hoek.

Chapter 5

TACTICS TO BE EVALUATED

5.1 INTRODUCTION

In Section 1.9, we have stated the aim of this dissertation to be as follows :

> *The aim of this dissertation is to show how counter missile tactics can be optimised. As a result the ship and missile will be modelled as generic concepts while the environment will be a chosen area of operations. The applicable methodology is to simulate the ship, missile and environment as well as the interactions between them. At the same time, the ship will be carrying out several missile counter measures. The methodology will then be to build a dynamic simulation model to optimise soft kill tactics by a generic FAC(M) against a generic surface-to-surface SSM in the chosen environment.*

We have developed the chosen environment in Chapter 2, the generic FAC(M) in Chapter 3 and the generic SSM in Chapter 4. From the FAC(M) model in Chapter 2, it is easy to see that it is feasible for the ship to carry out the following tactics:

- Increase speed to 30 knots.
- Alter course perpendicular to the missile's bearing.
- Jam the missile head radar to obscure the deployment of close range chaff.
- Deploy close range chaff.
- Deploy medium range chaff.

Note that the above-mentioned tactics are not a complete list of all possible tactics. In fact, we have chosen one possible tactic from the five main areas from which tactics can be derived. We assume that any combination of these actions, or the decision not to carry them out, can be considered a counter missile tactic. This leads us to 2^5 or 32 possible tactics. For the purposes of this dissertation we will number these tactics from A to E in the order that they appear in the list above.

If we elect to carry out Tactic A, then we denote it by

```
Tactic_A := TRUE;
```

and if we elect not to carry out the tactic, which could be a tactic in itself, we denote it by

```
Tactic_A := FALSE;.
```

This convention holds for all the tactics in the list. Furthermore, the FAC(M) will only consider these combinations of tactics once it has detected the oncoming missile, either by search radar or by its ESM equipment. In order to have a clearer picture of these tactics, we will examine them one at a time.

5.2 CHANGING SPEED

Tactic A is a tactic whereby the ship increases speed to 30 knots. The advantage of increasing speed is that the ship becomes more manoeuvrable at higher speeds and that it might be possible that the ship may inadvertently place itself outside of the missile's path. To what extent the latter is true, we will see from the simulation.

In Section 3.4.1 we have considered the initial speed of the ship to be in the interval [22,28] knots. If `Tactic_A := TRUE`, then the ship will increase speed as described in Section 3.4.3 to 30 knots immediately after the missile has been detected and the FAC(M)'s crew has acted upon that information, that is, the reaction time to commence the increase in speed has elapsed.

5.3 CHANGING COURSE

Course changes must, of necessity, also endeavour to achieve the minimising of the missile's hit probability. Two courses of action are worth mentioning. Firstly, the ship can alter course to minimise its own radar cross section in regard to the missile direction. Secondly, the ship can turn to a new course at right angles to the missile's bearing in order to increase the missile's angular tracking rate and to displace itself as far away as possible from its original position perpendicular to the missile's present flight path. In this case we assume that the missile is aimed at the ship's position at the time when the turn commences.

We choose the latter course of action for our simulation. Again, the turn to the new course will commence after the missile has been detected and the crew's reaction time has elapsed.

5.4 JAMMING THE MISSILE HEAD RADAR

If the missile is locked onto the FAC(M) and the close range chaff is not placed in the missile head radar's range gate, the deployment of close range chaff will not work as the missile head radar will not see the close range chaff in its range gate. In order to break the missile's lock in range, the missile head radar is jammed by the ship's jammer. The idea is to force the missile into accepting home-on-jam and as the missile head radar's range tracking values might not be extremely accurate, when the jamming ceases, the missile is forced to search and reacquire the target. The result is that the missile range gate depth will be at its maximum and the simultaneous appearance of the ship's radar reflective energy and that of the close range chaff in the range gate becomes very probable.

Thus, jamming would normally be used in conjunction with the deployment of close range chaff. However, in order to analyse the effects of jamming, it must be considered in conjunction with all the other possible tactics[1], that is, as part of all the possible 2^5 combinations of the five stated tactics. In order to accommodate this requirement, when jamming takes place where close range chaff is not intended to be fired, then the jammer will jam the missile head radar for three seconds after the missile has been detected and the reaction time for ECM measures has elapsed.

5.5 DEPLOYING CLOSE RANGE CHAFF

In order to deploy close range chaff, the flight time of the missile is of importance as the effects of the close range chaff will diminish over time because the chaff will disperse and eventually fall into the sea. Also, the close range chaff must be sufficiently close to the ship when the missile head radar detects the ship so as to ensure that both the close range chaff and the ship is in the missile head radar's range gate at the same time. However, the end result must be that the missile misses the ship. This is achieved by the expected difference in the ship's velocity vector and the chaff's velocity vector.

In order to achieve this state, the close range chaff must be deployed when the missile head radar's range gate is likely to detect the FAC(M) shortly. Given the Justice Radar Simulation results from Section 4.3.3, one can argue that it is prudent to fire the close range chaff when the missile is about 12 000 metres away. We assume that the responsible operator would launch the first close range chaff rocket within one second of the time required to ensure the first close range chaff bloom at 12 000 metres. Also, we assume that the missile range from the ship when the first close range chaff bloom appears is uniformly distributed in the interval [11690,12310] metres.

Furthermore, we assume that the launching equipment is such that the second close range chaff bloom will appear exactly one second after the first close range chaff bloom appears.

The FAC(M) can elect either to fire the prescribed close range chaff pattern or it can elect not to fire same.

[1] E.P. Box, W.G. Hunter and J.S. Hunter, *Statistics for Experimenters*. New York: John Wiley, 1978, pp 306-307.

5.6 DEPLOYING MEDIUM RANGE CHAFF

The aim of deploying medium range chaff is to entice the missile head radar to lock onto a false target sufficiently far away so that

- the ship and the chaff bloom never appear in the range gate at the same time;

- when the missile breaks lock on the chaff, it is too late for it to search and acquire the ship; and

- if the missile assumes the ballistic phase, the missile's heading would be such that the missile would miss the ship.

For this end, it follows that medium range chaff should bloom when the missile is further away than is prudent for close range chaff. Also, the missile head radar must preferably acquire the chaff when the missile head radar's maximum range does not extend to the ship. On the other hand, one can assume that the range of a medium range rocket is restricted.

Again, given the Justice Radar Simulation results from Section 4.3.3, one can argue that it is prudent to fire the medium range chaff when the missile is in excess of 12 000 metres away. We choose to fire the medium range chaff once the missile has been detected and after the appropriate reaction delay time. In the absence of any data to this effect, we assume the reaction delay time, r_d, to be such that

$$r_d \sim \text{weibull}(1.5, 4.25)^2.$$

5.7 SUMMARY OF POSSIBLE COMBINATIONS OF TACTICS

We have noted that 2^5 combinations of the five stated tactics as described in Section 5.1 exist. Figure 5.1 depicts all the possible combinations. Note that where a tactic is elected, that is, the tactic is `TRUE`, it is denoted in Figure 5.1 as "+" and where the tactic is not elected, that is, the tactic is `FALSE`, it is denoted in Figure 5.1 as "–"

This convention was adopted to accommodate Yates' Algorithm[3] to calculate the effects of the various tactics and the interactions between them. However, note that in later sections we denote our experiments by $\mathrm{E}e_1e_2e_3e_4e_5$ where

$$e_i = \begin{cases} 1 & \text{if the tactic is carried out,} \\ 0 & \text{if the tactic is not carried out.} \end{cases}$$

For example, the combination "++––+" translates into the experiment E11001.

---

[2] A.M. Law, and W.D.Kelton, *Simulation Modeling and Analysis* (*sic.*). 2 ed. New York: McGraw-Hill, 1991, p 333.
[3] Box, Hunter & Hunter, *op.cit.* pp 342-344.

Tactic A	Tactic B	Tactic C	Tactic D	Tactic E
−	−	−	−	−
+	−	−	−	−
−	+	−	−	−
+	+	−	−	−
−	−	+	−	−
+	−	+	−	−
−	+	+	−	−
+	+	+	−	−
−	−	−	+	−
+	−	−	+	−
−	+	−	+	−
+	+	−	+	−
−	−	+	+	−
+	−	+	+	−
−	+	+	+	−
+	+	+	+	−
−	−	−	−	+
+	−	−	−	+
−	+	−	−	+
+	+	−	−	+
−	−	+	−	+
+	−	+	−	+
−	+	+	−	+
+	+	+	−	+
−	−	−	+	+
+	−	−	+	+
−	+	−	+	+
+	+	−	+	+
−	−	+	+	+
+	−	+	+	+
−	+	+	+	+
+	+	+	+	+

Figure 5.1: Possible combinations of the five stated tactics

5.8 PERFORMANCE MEASURES

Once all the possible combinations of the five stated tactics have been performed, it is necessary to find what the effects of and interactions between the five tactics are. For that end, we must choose some performance measures to allow meaningful analysis. Inherently, the aim of our simulation is to determine which combination of tactics will minimise the missile's hit probability, that is,

$$\text{Minimise } P(\text{Hit})$$

or, alternatively, to maximise the missile's miss distance, that is,

Maximise d_m.

In order to handle the two measures of performance, two variables are set to zero when the global variables with respect to the overall simulation are initialised, that is,

```
Mhit         := 0;
MissDistance := 0;
```

At the end of every replication, if the missile has hit the FAC(M), `Mhit` is incremented by one. At the same time `MissDistance` is incremented by the actual measured miss distance for that replication. At the end of the simulation, `MissDistance` is divided by the number of replications to obtain the mean miss distance. We can consider `Mhit`, or M, as

$$M = \sum_{i=1}^{n} h_i$$

where

$$h_i = \begin{cases} 1 & \text{if} \quad \text{missile hit} \\ 0 & \text{if} \quad \text{missile missed.} \end{cases}$$

Furthermore, we consider `MissDistance`, or d_m, as

$$d_m = \frac{1}{n}\sum_{i=1}^{n} x_i$$

where x_i is the miss distance of the ith replication and d_m is thus the mean miss distance observed.

5.8 VALIDATION OF CHOSEN TACTICS

The tactics described above were scrutinised by the South African Navy's current Strike Craft[4] Squadron Commander and four of his Officers Commanding. We reached the conclusion that, although the chosen tactics were not necessarily representative of their current prescribed tactics, they were necessary and sufficient for the purposes of our generic models. Also, we were satisfied that the chosen tactics were fully developed. Therefore, we assume the chosen tactics to be requisite and, in turn, we assume them to be valid.

[4] The SA Navy has designated their FAC(M) as Strike Craft.

Chapter 6

THE SIMULATION MODEL

6.1 INTRODUCTION

In order to explain the simulation model, we shall make use mainly of flow diagrams and the following associated conventions[1]:

- We shall use a top-down approach.
- The highest level in the simulation hierarchy shall be denoted as a level 0 process.
- The sub-processes contained in the level 0 flow diagram will be denoted numerically as processes 1, 2, ..., n, that is, we shall refer to those as processes 1, 2, ..., n.
- In turn, the further breakdown of processes 1, 2, ..., n will be denoted sub-processes $j.1$, $j.2$, ..., $j.m$ where $j \in \{1, 2, ..., n\}$, etcetera. For example, in line with the previous convention, we will refer to the first sub-process of process 1 as sub-process 1.1 and, in turn, to the first sub-sub-process in sub-process 1.1 as sub-process 1.1.1.

Furthermore, this chapter shall explain the model to a level where the algorithms and source code given in the previous chapters are placed in their proper context. Therefore, the depth of the discussion might vary for various parts of the simulation model.

Also, note that the flow diagrams used will not necessarily explain the use of a particular Pascal construct. For example, for the Pascal construct

```
for counter := 1 to n do
   begin
      <<Pascal statements>>
   end;
```

we may place the decision point of whether the variable, `counter`, contains the value n, either before or after the execution of the applicable action block in the flow

[1] M.J. Powers, P.H. Cheney and G. Crow, *Structured Systems Development*. 2 ed., Boston: Boyd & Fraser, 1990, pp 91-96.

diagram. The only constraint is that the logic of the flow diagram and the logic within the Pascal construct must be identical.

6.2 OVERALL CONCEPT

The overall concept model is depicted in Figure 6.1. The first action at the commencement of the simulation is to initialise the variables that appertain to the whole of the simulation.

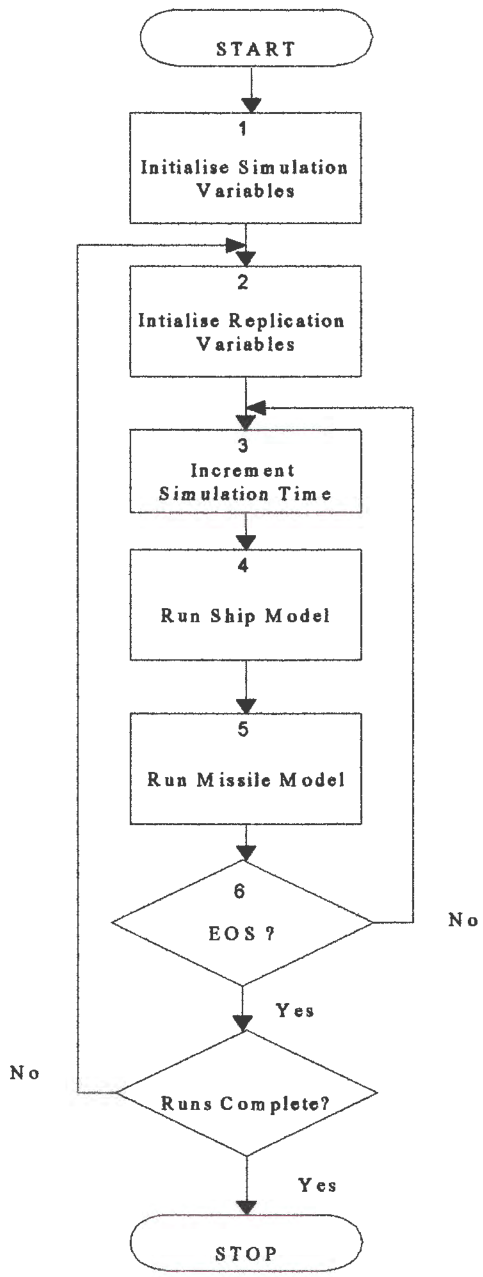


Figure 6.1: Level 0 flow diagram

Thereafter, the various simulation replications can be executed. Again, the first action within every replication is to initialise the values of particular variables that pertain to the replication in progress.

Note that, amongst other variables, the initial positions of both the ship and the missile are fixed in the initialisation of the variables that appertains to a particular replication. The second step in any particular replication of the simulation is to increment the time elapsed since the start of the replication. Thereafter, the ship and the missile's respective states are updated for the new replication elapsed time. For brevity's sake, we shall, in the remainder of this discussion, refer to the elapsed replication time simply as the *time* or *replication time*. As the time increment corresponds with the pulse interval of the missile head radar, it follows that one missile head radar pulse is simulated at every time increment.

On completion of the states of both ship and missile being updated, we test to determine whether the missile has

- hit the target;
- missed the target; or
- whether neither of the two events have already occurred.

If the missile has either hit or missed the target, the replication is terminated and the necessary performance measures are updated accordingly. Thereafter, the model checks to ascertain whether the simulation is complete. If it is the case, the program reports on the performance measures and terminates the simulation; otherwise, it simply orders a new replication to be executed. If the missile has neither hit or missed the target, the replication continues by incrementing the time and updates the ship and missile states respectively.

The level 0 flow diagram in Figure 6.1 represents the main section of the Pascal program that is in Appendix G at page G-43. Note that a report file is opened in the main program to facilitate a text output from the program. Also, the final result is displayed on the screen. As these program instructions are not central to the simulation, they were omitted from Figure 6.1.

6.3 INITIALISING THE SIMULATION VARIABLES

The initialisation of global variables in regard to the overall simulation refers to process 1 in Figure 6.1 and is aimed at inserting fixed values into the simulation for all replications. This procedure is in Appendix G at pages G-38 to G-40 and initialises the following:

- Sets up random number seeds for all sets of random numbers.
- Sets up the simulation time step or simulation interval.
- Links sets of random numbers to particular variables.
- Reads the WIND.DAT file and initialises the distributions for wind direction and wind speed.

- Reads the RADAR.DAT file and initialises the distribution for search radar detection range.

- Reads the RCSPOS.DAT file and initialises the ship's radar reflective positions.

- Reads the RCSMAG.DAT file and initialises the radar cross section matrix in regard to radar cross section for various relative bearings.

- Reads the ALTCO.DAT file and initialises the ship's turning rate parameters.

- Sets the number of hits observed to zero.

- Sets the mean miss distance to zero.

Once these variables are initialised, the model is ready to commence with the ordered number of replications.

6.4 INITIALISING THE REPLICATION VARIABLES

The initialisation of global variables in regard to particular replications of the simulation refers to process 2 in Figure 6.1 and is aimed at inserting fixed values into the particular replication to enable the unique parameters of that replication. Note that there exist twenty stochastic processes and the same number of Boolean variables that must be initialised for every replication. This procedure is named `procedure InitialiseGlobalVariablesRepilication` and is found in Appendix G at pages G-36 to G-38 and it initialises the following:

- Sets the end-of-replication variable to false.

- Sets the replication run time to zero.

- Initialises the range and angle error stacks to empty.

- Generates a random wind vector.

- Generates a random current vector.

- Generates a random air temperature.

- Sets the FAC(M)'s initial position.

- Generates a random FAC(M) course and speed and initialises the ship's Cartesian course and speed vector.

- Sets up the FAC(M)'s initial logic states in regard to the following:

  Alteration of speed not ordered.
  Alteration of course not ordered.
  Missile head radar not detected.
  Jammer off.
  Missile not detected by search radar.
  Reaction time to carry out ordered alterations of speed and course not elapsed.
  Turn to starboard.
  Fire close range chaff to starboard.
  Execution of tactics not ordered.
  Do not fire close range chaff.
  Do not fire medium range chaff.

- Generates a random search radar detection range for an oncoming missile.

- Generates a random initial position for the missile and initialise missile's Cartesian grid position.

- Generates a random missile course and calculate missile speed. Thereafter, initialises the missile's Cartesian course and speed vector.

- Sets up the missile's initial logic states in regard to the following:

  Search phase not ordered.
  Acquisition phase not ordered.
  Tracking phase not ordered.
  Home-on-jam not ordered.
  Pursuit phase not ordered.
  Proportional navigation not ordered.

- Sets missile head radar range gate range such that the ballistic logic is inhibited.

- Calculates missile's relative bearing in regard to the FAC(M) and initialises relevant radar cross section data. Also, sets relative bearing test value.

- Sets radar cross section points and magnitudes in Cartesian grid.

- Generates FAC(M)'s ESM random detection range on missile head radar.

- Initialises chaff list to zero.

- Sets number of close range chaff rockets fired to zero.

- Sets number of chaff bundles fired to zero.

- Sets range test to arbitrarily chosen large value.
- Generates random delay times for tactics C, D and E.

Once these variables are initialised, the model is ready to commence with a particular replication.

6.5 INCREMENTING SIMULATION TIME

Process 3 in Figure 6.1 increments the replication time elapsed. The reason for keeping track of the time elapsed is to enable the execution of discrete events in parallel with the continuous simulation of the FAC(M) and SSM movements and other inherent actions.

For example, if a change of speed will be ordered, it would be a discrete event that is dependent on whether the missile has been detected and the FAC(M) crew's reaction time which can be simulated by a stochastic process. We denote the replication time elapsed by T. Let us assume that the missile is detected at time T_i and that the crew's reaction time is t_r. The time at which the discrete event, that is, the beginning of the speed change, must happen will be at time $T_j = T_i + t_r$. Now, by testing at subsequent iterations of the ship model whether the time specified for the discrete event is equal or less than the simulation time, that is, whether $T \geq T_j$, the discrete event can be inserted at the appropriate time.

As the time increments in our model is sufficiently small, that is, the time increment is 0.002 seconds, any errors resulting from the situation where $T > T_j$ will be in the interval (0,0.002) seconds and can, in regard to this simulation model, be regarded as inconsequential.

The source code uses the variable name `Stime` for T and it is incremented by the simple Pascal construct below.

```
{---------------------------}
{ Increment simulation time }
{---------------------------}
Stime := Stime + Dtime;
```

Note that `Dtime` is initialised as a simulation global variable (See Section 6.3) and the Pascal source code is simply

```
{-----------------------------}
{ Set up simulation time step }
{-----------------------------}
Dtime := 1/prf;
```

where `prf` is the pulse repetition frequency of the missile head radar.

6.6 RUN SHIP MODEL

The sequence of the ship model is depicted in Figure 6.2. Note that this sequence is enabled within the program by `procedure ShipModel` and can be found in Appendix G at pages G-33 to G-36. Also, note that this diagram is not a flow diagram but only a sequential representation of the activities within the ship model.

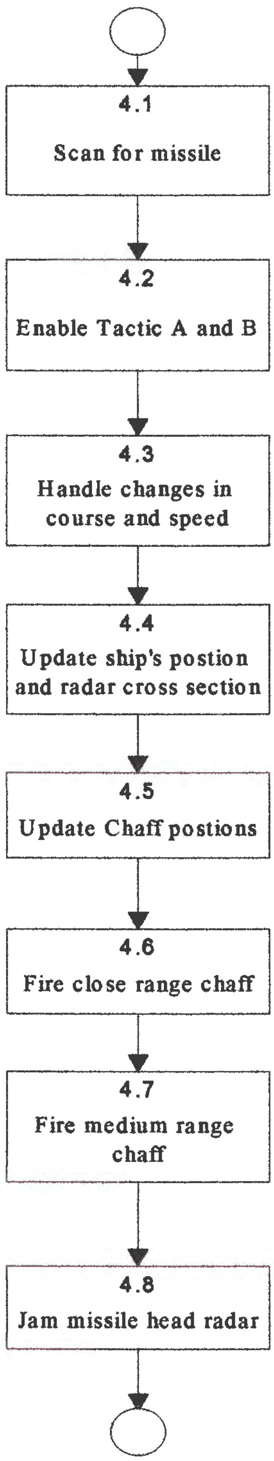


Figure 6.2: Run Ship Model

6.6.1 SCAN FOR MISSILE

The Scan for Missile sub-process is depicted in Figure 6.3.

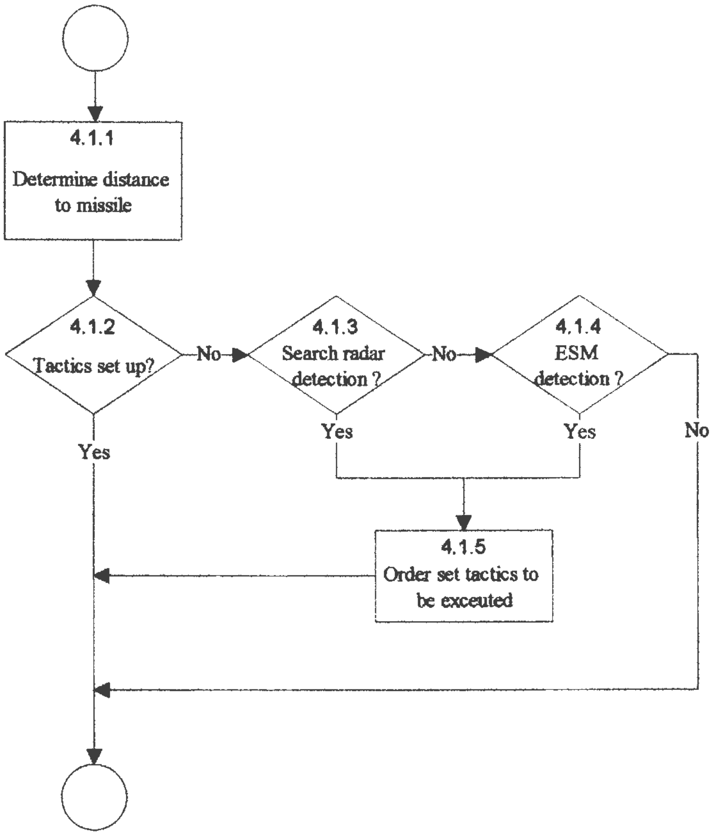


Figure 6.3: Scan for missile sub-process

From Figure 6.3, the sub-process is self-explanatory. However, in order to execute sub-process 4.1.5, `procedure ExecuteTactics` are called. This procedure is documented in Appendix G on pages G-30 to G-33. Within this procedure, the tactics to be evaluated are set up for execution by the program. This procedure is explained in Figure 6.4.

Note that the missile bearing and range must be determined before any evaluation in terms of the ordered tactics can be executed. For ease of reference, the various tactics, when set to TRUE, are repeated below.

- Tactic A: Increase speed to 30 knots.
- Tactic B: Alter course to maximise the SSM's angular tracking rate.
- Tactic C: Jam the missile head radar.
- Tactic D: Deploy close range chaff.
- Tactic E: Deploy medium range chaff.

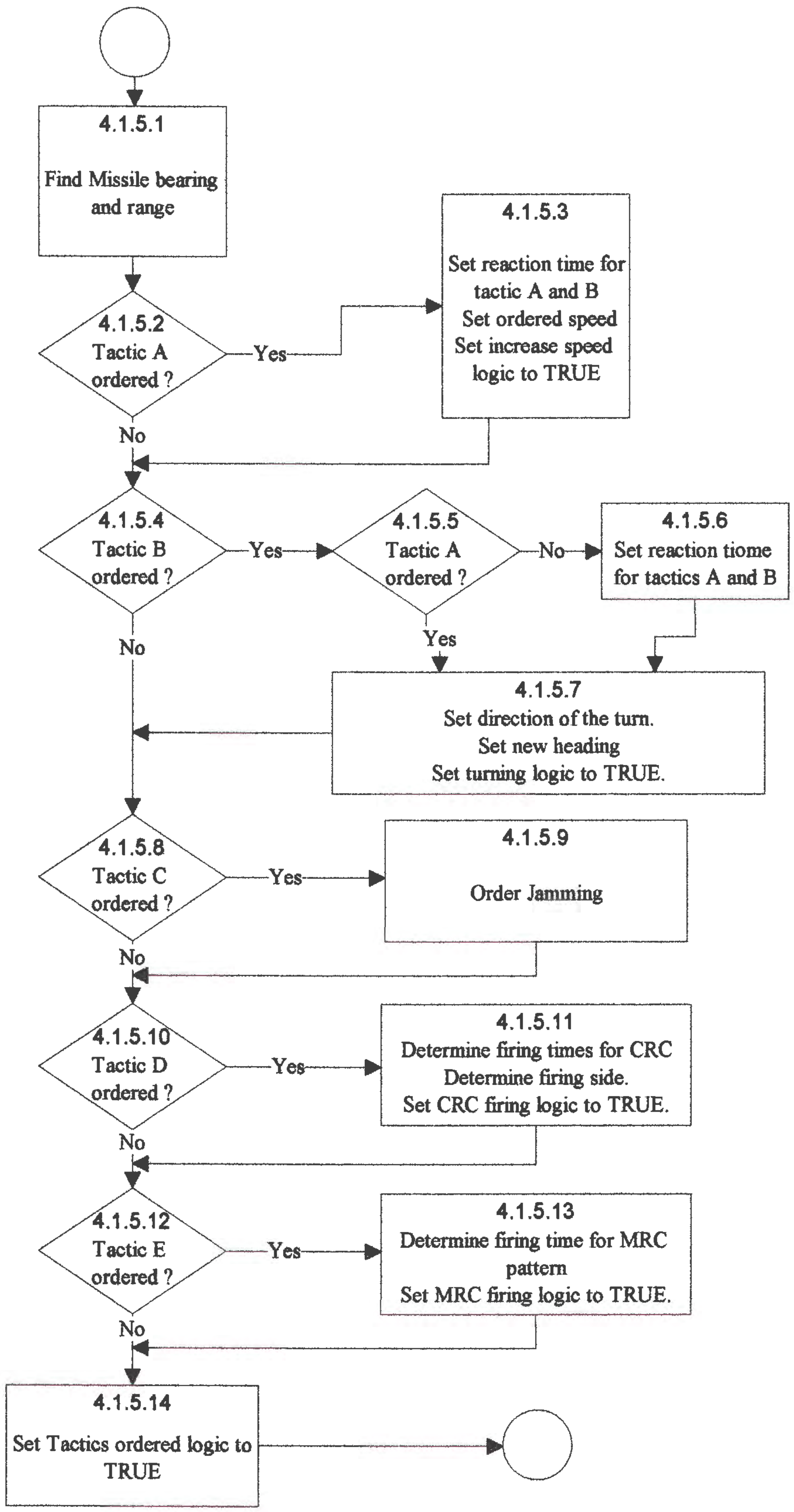


Figure 6.4: Order set tactics to be executed

6.6.2 ENABLE TACTIC A AND TACTIC B

In order to enable the execution of both, or any one of, Tactic A and Tactic B, if they were ordered, the following simple Pascal source code was used.

```
{-------------------------------------}
{ Enable Tactics at appropriate time }
{-------------------------------------}
if not(ReactionTimeElapsed) AND (Stime > TactABTime) then
   begin
      ReactionTimeElapsed := TRUE;
   end;
```

This fragment allows the insertion of the discrete events, increase speed to 30 knots and alter course to maximise the SSM's angular tracking rate within 0.002 seconds of the appointed time in our continuous simulation. (See Section 6.5) Note that `Stime` is the time elapsed for the particular replication whilst `TactABTime` is time that the two discrete events must commence.

6.6.3 HANDLE CHANGES IN COURSE AND SPEED

Once the turn or the increase in speed has commenced, this is reflected at every iteration of the ship model. This aspect is elucidated in Figure 6.5.

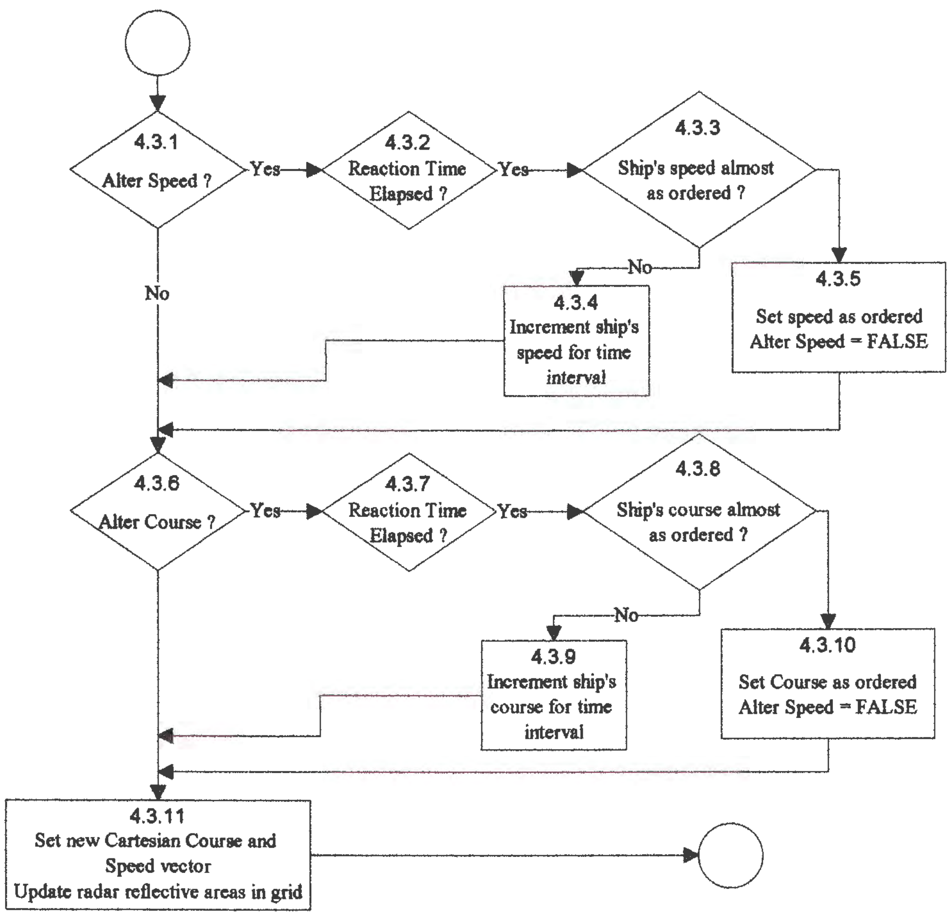


Figure 6.5: Handle changes in course and speed

The term almost in sub-processes 4.3.3 and 4.3.8 is regulated by setting a constant sufficiently small to test whether the desired result has been reached. With floating point operations, it is necessary to do such a test as the probability that outcome of the operation to change speed or course will match the desired result exactly, is relatively small. Thus, a test value, such that the difference between the desired result and the outcome of the floating point operation can be compared to establish whether the course or speed change has been completed, is called for. In Pascal engineering notation, these test values were determined to be

```
SpeedDiff  =  1E-5,
```

and

```
CourseDiff =  1E-3.
```

6.6.4 UPDATE SHIP'S POSITION AND RADAR CROSS SECTION

The ship's position is incremented by subtracting the ship's total vector, that is the ship's course and speed as well as the influence of the current on the ship, to the missile's position. This allows for the ship to remain in position (0.0) in our Cartesian grid whilst the missile is moving relative to the ship. This convention was decided upon in order to minimise the amount of radar reflective areas that must be updated at every iteration of the ship model. Therefore, when the ship is on a steady course, the radar reflective areas need not be updated in regard to their positions in the grid. This is achieved by the Pascal source code given below.

```
SSM.posn[1] := SSM.posn[1] - (ship.move[1] + CurrentV[1])/prf;
SSM.posn[2] := SSM.posn[2] - (ship.move[2] + CurrentV[2])/prf;
```

Now, the radar cross section of the ship changes as the relative bearing of the missile changes during the replication. In order to facilitate this changing of the radar cross section as the relative bearing changes, we set some test value equal to the relative bearing of the missile to the ship at a particular iteration of the ship model. The test value is stored in the variable `RCSTest`. Also, the relative bearing of the missile to the ship, `RCSIndex`, is rounded to the nearest 10° as the data available to us estimates the radar cross section for every 10° of relative bearing change. (See Section 3.3)

The `SSMRelBrg` procedure calculates the value of `RCSIndex` and the `ShipRCSMag` procedure sets the present values of the 17 radar reflective points. The Pascal source code for this part of the sub-process is given directly.

```
{---------------------------------------}
{ Update ship's radar reflective areas }
{---------------------------------------}
SSMRelBrg(SSMBrg,degree(ship.head),RCSIndex);
if RCSIndex<>RCSTest then
   begin
      ShipRCSMag;
   end;
```

6.6.5 UPDATE CHAFF POSITIONS

If chaff has been fired, the positions of the chaff bundles must also be updated at every iteration of the ship model. Again, these positions must be updated so that the chaff moves relative to the ship. Therefore, we add the wind vector to the chaff positions and subtract the total ship vector, that is, the resultant vector for course, speed and the effects of the current, from them. The Pascal source code to execute this sub-process, follows directly below.

```
{---------------------------}
{ Increment chaff positions }
{---------------------------}
if ChaffCount > 0 then
   begin
      for chaff_i := 1 to ChaffCount do
         begin
            chafflist[chaff_i,1] := chafflist[chaff_i,1] +
                                   (WindV[1] - ship.move[1] -
                                    CurrentV[1])/prf;
            chafflist[chaff_i,2] := chafflist[chaff_i,2] +
                                   (WindV[2] - ship.move[2] -
                                    CurrentV[2])/prf;
         end;
   end;
```

6.6.6 FIRE CLOSE RANGE CHAFF

This sub-process regulates the firing of close range chaff. It fires one close range chaff rocket at the appointed time and thereafter it will fire another every second until the ordered number of rockets had been fired. Again, this procedure allows for discrete events, that is, the firing of a number of close range chaff rockets, to be incorporated in the continuous simulation replication. The algorithm is similar to the one to incorporate changes of speed and course into the replication and the Pascal source code is given directly. The variables used are self-explanatory.

```
{------------------------------------------}
{ Fire close range chaff at appointed time }
{------------------------------------------}
if CRCdeploy AND (STime>CRCfTime[CRCindex]) then
   begin
      FireCRC;
      CRCindex := CRCindex + 1;
      if CRCindex > maxCRC then
         begin
            CRCdeploy := FALSE;
         end;
   end;
```

Note that the procedure `FireCRC` fires the rocket. This procedure was dealt with in Section 3.6.3 and that detail will not be repeated here.

6.6.7 FIRE MEDIUM RANGE CHAFF

The only significant difference in the method to fire medium range chaff as opposed to close range chaff is that all the medium range chaff rockets are fired simultaneously. The Pascal source code is given directly.

```
{-------------------------------------------}
{ Fire medium range chaff at appointed time }
{-------------------------------------------}
if MRCdeploy AND (STime>MRCfTime) then
   begin
      FireMRC;
      MRCdeploy := FALSE;
   end;
```

Again, note that the procedure `FireMRC` fires all the rockets. This procedure was dealt with in Section 3.6.4 and that detail will not be repeated here. Figure 6.6 depicts the medium range chaff firing logic.

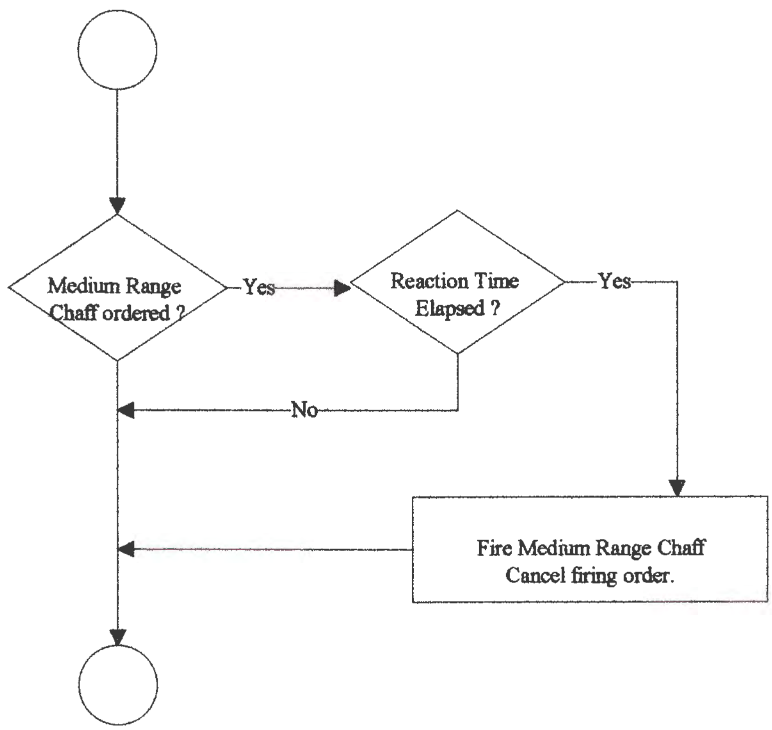


Figure 6.6: Medium range chaff firing logic

6.6.8 JAM MISSILE HEAD RADAR

The jam missile head radar sub-process regulates the jamming logic. The logic is depicted in Figure 6.7. Once jamming is ordered by sub-process 4.1.5.9, the ship model accepts the command and it switches on the jammer during the same iteration of the ship model.

In subsequent iterations of the ship model it tests whether the jamming order was given and whether the jammer is switched on. If both conditions are true, the sub-process tests whether Tactic D, that is, the tactic to fire close range chaff has been ordered.

If Tactic D was ordered, the sub-process tests whether all the close range chaff rockets were fired. In our simulation, this corresponds to the fact that all the close range chaff bundles have bloomed. If so, the jammer is switched off and the jammer command is cancelled.

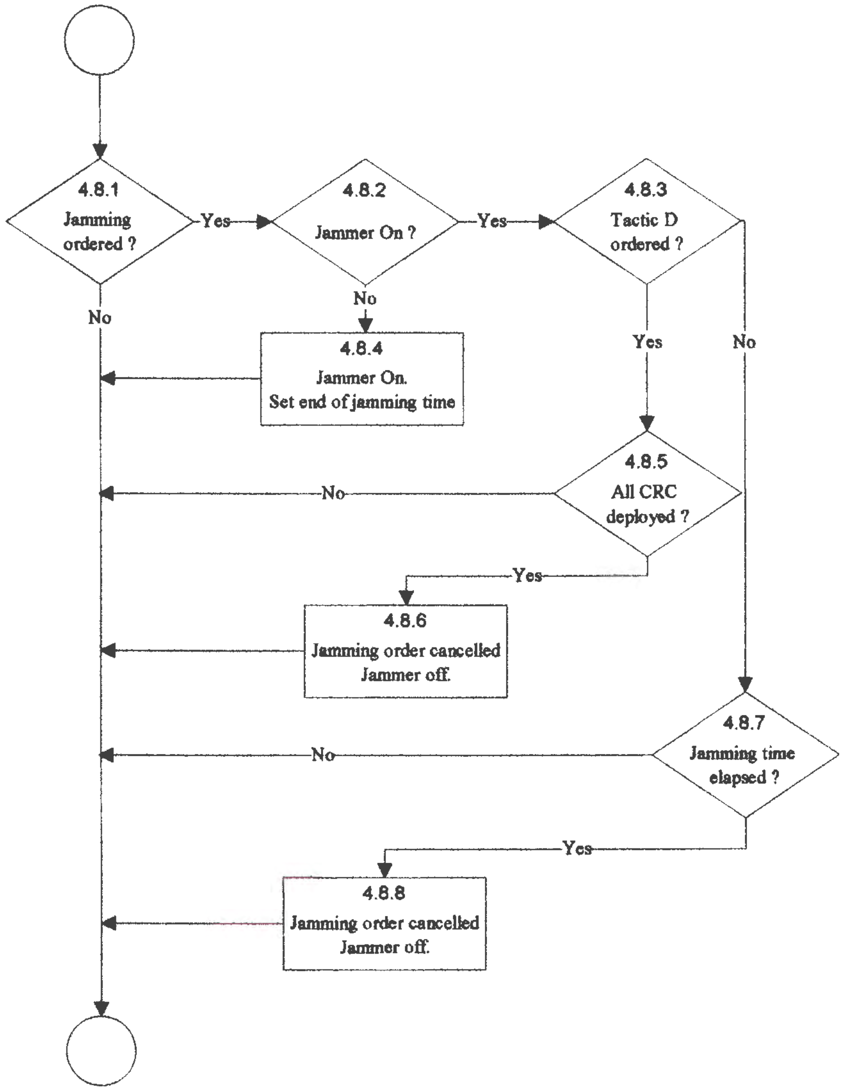


Figure 6.7: Jamming logic

In the case where Tactic D was not ordered, the sub-process will check to see whether the jamming time has elapsed. If so, it will switch off the jammer and cancel the jamming order.

From Section 3.6.2, for our simulation method the ship model depends only on the logic JAMMER ON or JAMMER OFF. The effect of the jammer is simulated within the SSM model.

6.7 RUN MISSILE MODEL

The sequence op the missile model is depicted in Figure 6.8. This sequence is enabled in the program by `procedure SSMModel` and can be found in Appendix G at pages G-28 to G-30. Again, note that this diagram is not a flow diagram but only a sequential representation of the activities within the missile model.

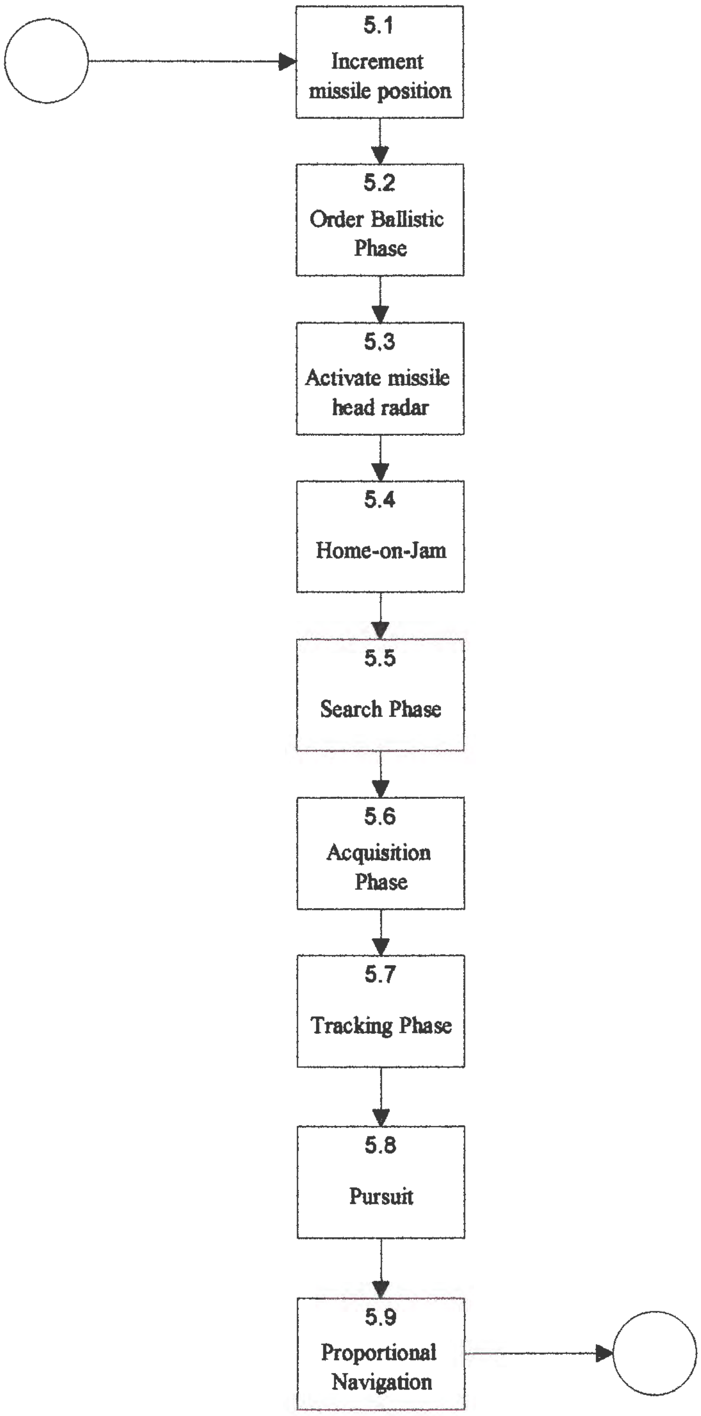


Figure 6.8: Run Missile Model

6.7.1 INCREMENT MISSILE POSITION

The first step in any iteration of the missile model is to increment the missile's position in the grid to allow for the missile's movement for the preceding pulse interval. This is achieved by the Pascal source code that follows directly.

```
{------------------------------}
{ Increment missile's position }
{------------------------------}
SSM.posn[1] := SSM.posn[1] + (SSM.move[1] + WindV[1])/prf;
SSM.posn[2] := SSM.posn[2] + (SSM.move[2] + WindV[2])/prf;
```

Note that the missile's speed vector and the wind vector are added to establish the missile's movement in the grid.

6.7.2 ORDER BALLISTIC PHASE

If the missile is within 150 metres of the target, it assumes the ballistic phase (See Section 4.7). The ordering of the ballistic phase is achieved by the Pascal source code that follows directly.

```
{----------------------------------------------------}
{ Assume ballistic flight path for final 150 meters }
{----------------------------------------------------}
if NOT(BALLISTIC) and (SSM.gate.range<150.0) then
   begin
      BALLISTIC := TRUE;
   end;
```

6.7.3 ACTIVATE MISSILE HEAD RADAR

The activation of the missile head radar was discussed in Section 4.3.3. The ordering of this discrete event is achieved by the Pascal source code that follows directly.

```
{---------------------------}
{ Activate MHR if required }
{---------------------------}
if NOT(MHR)AND
  (sqrt(sqr(SSM.posn[1])+sqr(SSM.posn[2]))<MHRactRange) then
   begin
      MHR    := TRUE;
      SEARCH := TRUE;
      SetSearchGate(MaxSAngle,LSearchL);
   end;
```

Note that when the missile head radar is activated, the missile automatically assumes the search phase and that the missile head radar is positioned to commence the search. The procedure to implement the latter instruction is in Section 4.5.

6.7.4 HOME-ON-JAM

Once the missile detects the jammer operating against it, the missile issues a home-on-jam (HOJ) command. This is achieved by the Pascal source code that follows directly.

```
{----------------------------}
{ Home on Jam when required }
{----------------------------}
if JAMMER AND NOT(HOJ) then
   begin
      HOJ := TRUE;
   end;
```

Similarly, once the missile detects the absence of the jammer, the home-on-jam command is cancelled by the Pascal source code that follows directly.

```
if HOJ AND NOT(JAMMER) then
   begin
      HOJ := FALSE;
   end;
```

6.7.5 SEARCH PHASE

The search sub-process is depicted in Figure 6.9. Note that the missile range gate has already been positioned by sub-process 5.3. The simulation of the missile head radar's pulse in the search phase, complete with the logic that regulates it, is the objective of this sub-process.

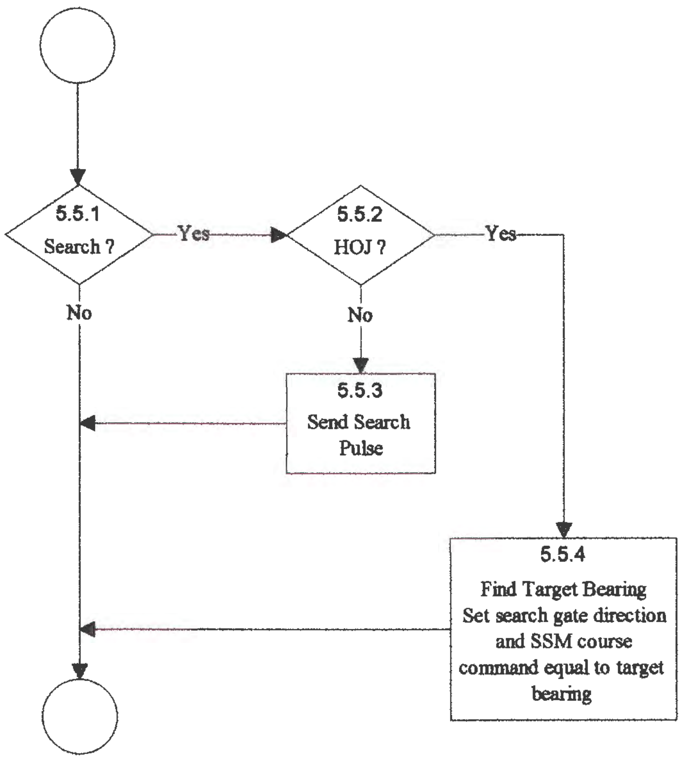


Figure 6.9: Search phase logic

Sub-process 5.5.3, that is, the send search pulse logic and sub-process 5.5.4, that is, the home-on-jam logic, will now be more fully explained. The search pulse logic is depicted in Figure 6.10.

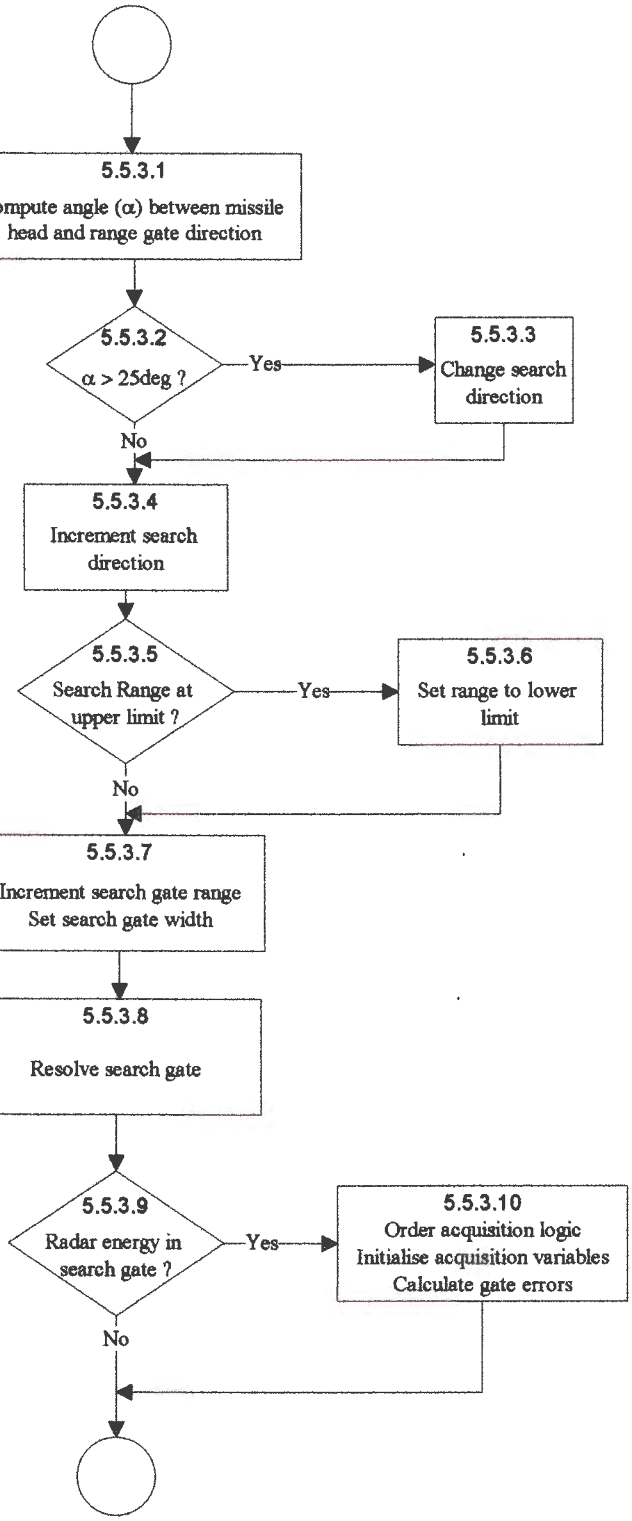


Figure 6.10: Send search pulse logic

Note that, for every iteration, firstly, the search gate is positioned in accordance with the search parameters from Section 4.5, then the area covered by the gate is checked for radar reflective energy emanating from it. If radar reflective energy is discovered in the gate, the missile generates the acquisition command. However, it does not cancel the search command immediately. In Section 6.7.6, the reason for this convention will be explained.

For the home-on-jam logic in the search phase, we note that the simulation is simplified by, instead of resolving the search gate, assuming that radar energy will be present in the gate and that the missile will track the jammer accurately. As a result, the actions listed in sub-process 5.5.3.10 are executed by the simulation.

Now, the algorithm used for resolving the search gate, that is, sub-process 5.5.3.8, is also used for resolving the range gate in the acquisition and tracking phases. This method was discussed in Section 4.4. The flow diagram for the algorithm to resolve the range gate is in Figure 6.11.

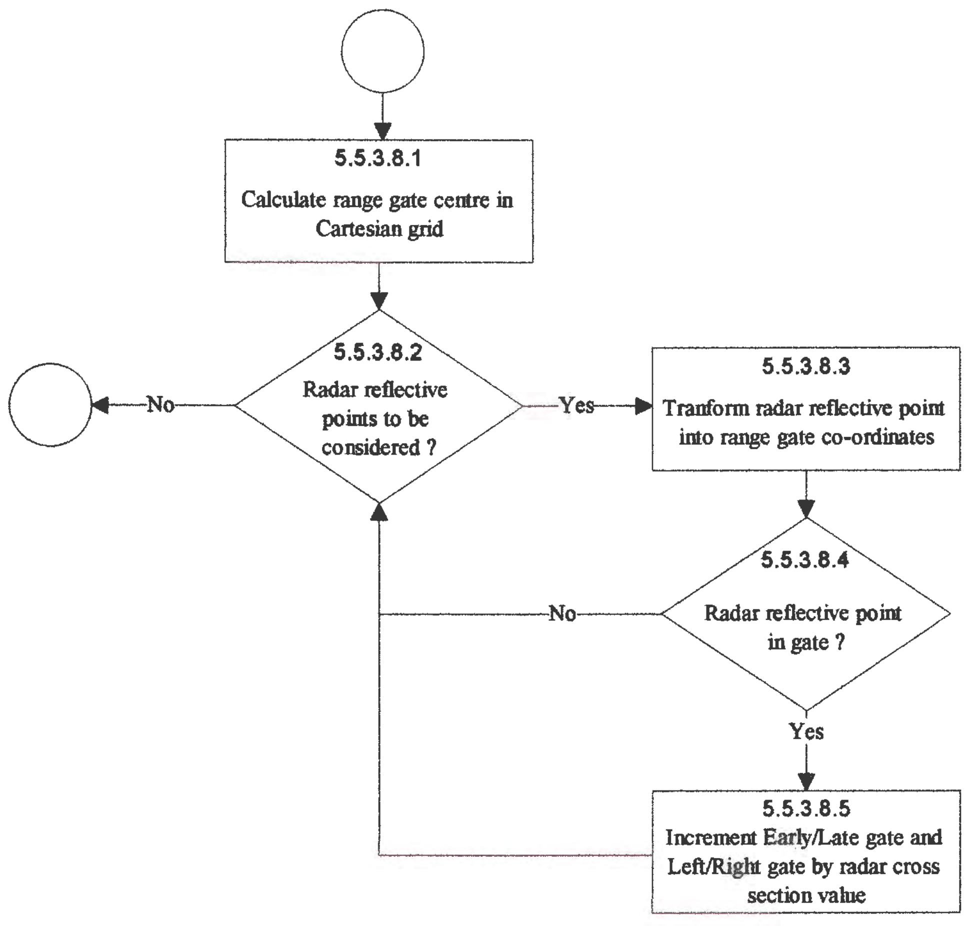


Figure 6.11: Resolving the range gate

6.7.6 ACQUISITION PHASE

During the acquisition phase, the depth of the range gate is reduced to 40 metres and thereafter, the absolute value of the range rate, $\dot{r}$, and the angular movement of the range gate's centre position, $\dot{\lambda}$, is calculated. Once $\dot{r}$ and $\dot{\lambda}$ is known, the missile is ordered into the tracking phase. (See Section 4.6) The logic for the acquisition phase is explained in Figure 6.12.

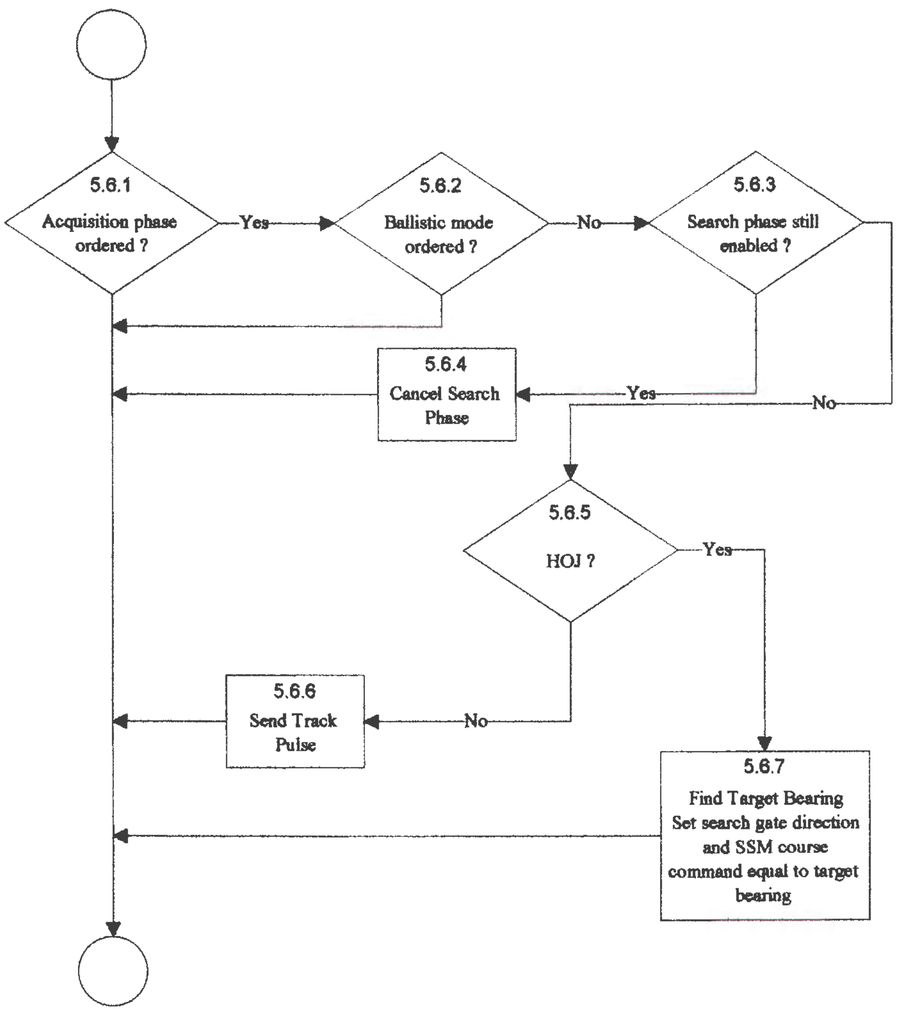


Figure 6.12: Acquisition phase logic

Note that sub-process 5.6.3 inhibits the sending of a tracking pulse if a search pulse was already sent in a particular iteration of the missile model. This is necessary as the model has not yet been incremented for the next time interval. The track pulse which is ordered in sub-process 5.6.6, uses the same source code as sub-process 5.7. To accommodate the contraction of the range gate to 40 metres, the Pascal source code that follows directly, is used.

```
{-------------------------------}
{ Set Range Gate for next pulse }
{-------------------------------}
if ACQUIRE then
   begin
      AcqGate   := AcqGate * AcqFactor;
      if AcqGate < 40.0 then
         begin
            AcqGate := 40.0;
         end;
   end
else
   begin
      AcqGate := 40.0;
   end;
```

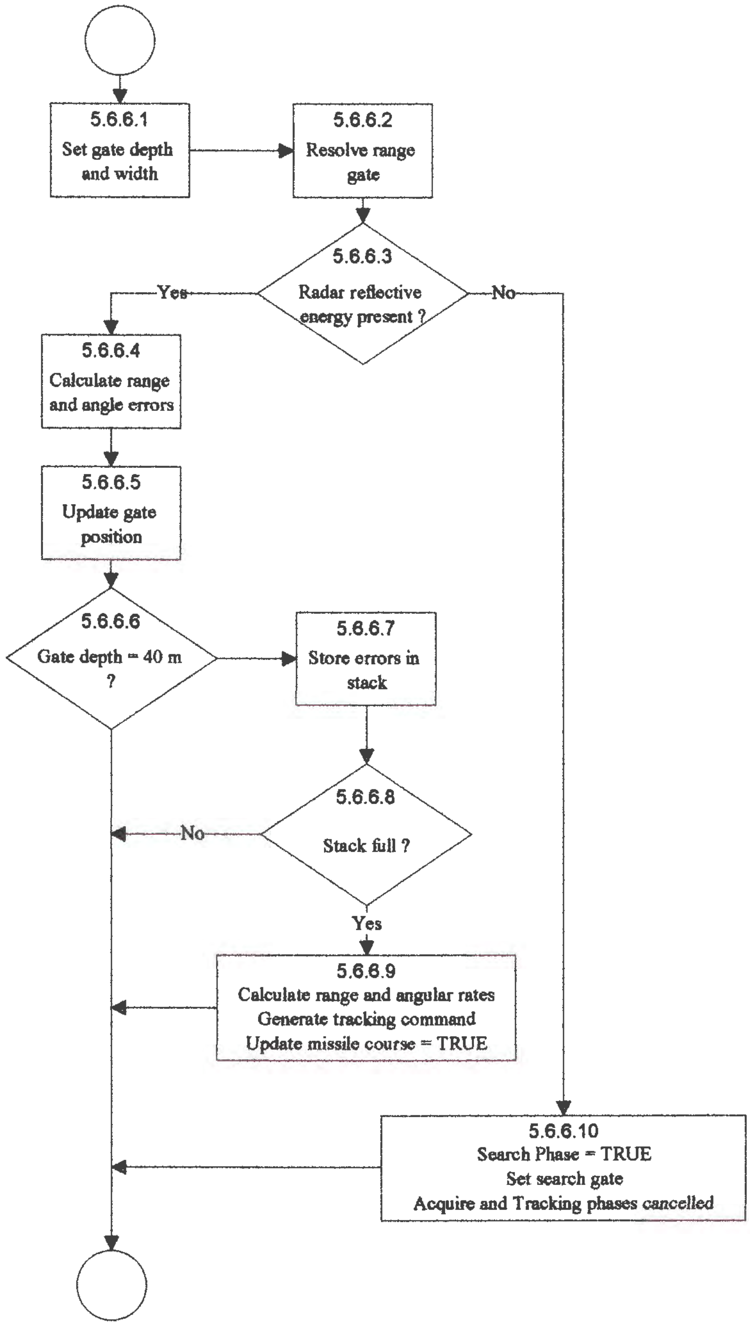


Figure 6.13: Send tracking pulse

Also, note that sub-process 5.6.7 and sub-process 5.5.4 are identical. If home-on-jam is ordered, the model assumes that radar energy will be present in the gate and that the missile will track the jammer accurately.

Now, the tracking pulse is depicted in Figure 6.13. The Pascal source code for the generation of the tracking command in sub-process 5.6.6.9 are given below.

```
{-------------------------------}
{ If SSM in acquisition phase   }
{ SSM advance to tracking phase }
{-------------------------------}
if ACQUIRE then
   begin
      LOCKED := TRUE;
   end;
```

6.7.7 TRACKING PHASE

The tracking phase logic is depicted in Figure 6.14

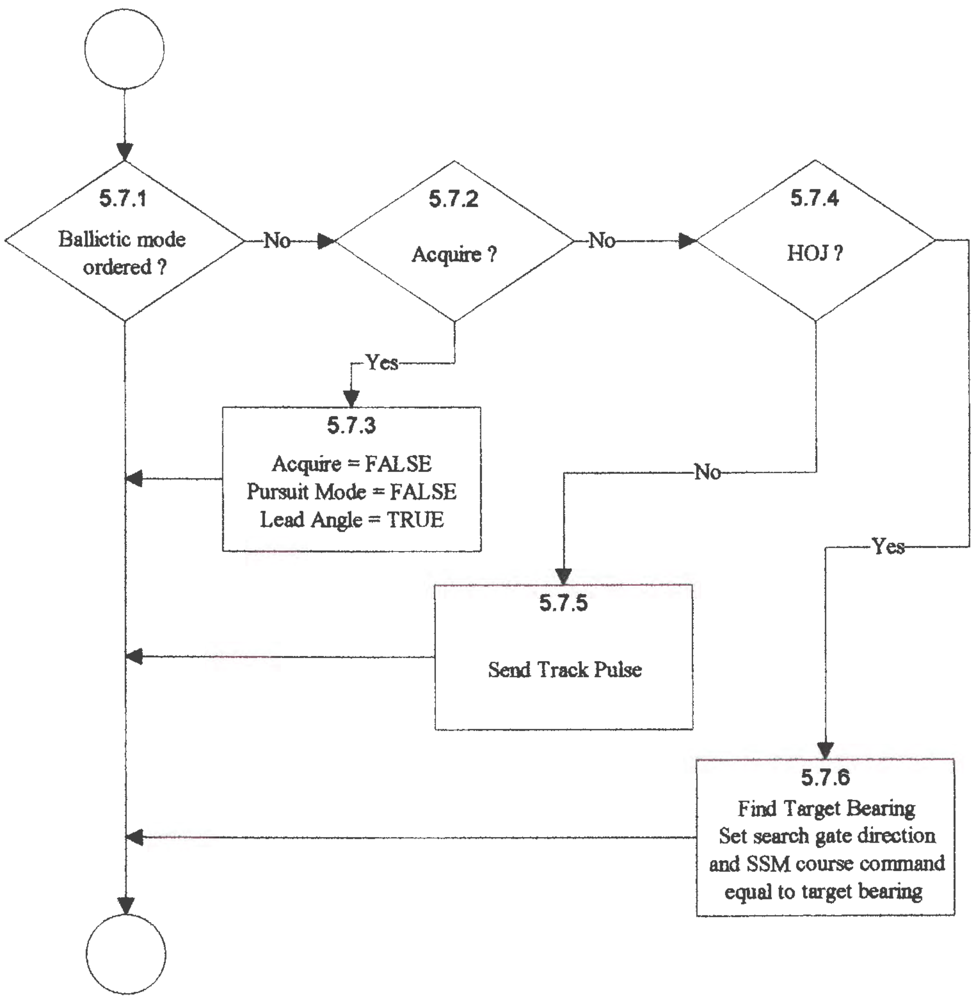


Figure 6.14: Tracking phase logic

Again, note that sub-process 5.6.7 and sub-process 5.5.4 are identical to sub-process 5.7.6. Also, sub-process 5.7.2 inhibits the track pulse from being sent again in the same iteration of the missile model. This will otherwise be the case when the track pulse is sent in the acquisition phase and on completion the tracking phase is ordered again.

6.7.8 PURSUIT HOMING

Whilst the missile is in the search phase or if the home-on-jam command is in force, the missile will pursue the target by adopting pursuit guidance. The target bearing is assumed to be the direction in which the missile head radar is pointing, that is, the bearing of the centre of the range gate from the missile head radar antenna. The Pascal source code to implement the pursuit mode follows directly.

```
if PURSUIT AND NOT(SEARCH) then
   begin
      SSM.aim := SSM.gate.direction;
      SSMAlterCourse;
   end;
```

Note that the direction command is stored in `SSM.aim`. The procedure called to effect the alteration of the missile's course simply allows the missile to alter course in the right direction at 70°/s. The alter course logic is depicted in Figure 6.15.

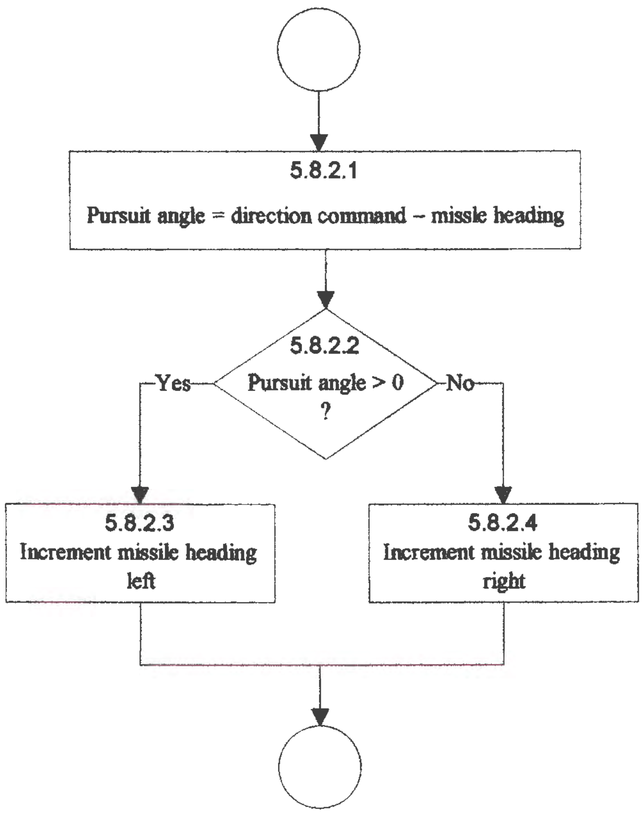


Figure 6.15: SSM alter course logic

6.7.9 PROPORTIONAL NAVIGATION HOMING

The proportional navigation homing method was described in Section 4.8. The implementation is straightforward and the Pascal source code is given directly.

```
if LEADANGLE AND NOT(BALLISTIC)  then
   begin
      if UpdateAim then
         begin
            TimeGain  := SSM.gate.range/abs(MeanRVector);
            SSM.aim   := SSM.aim + LAGain*TimeGain*MeanAVector;
            UpdateAim := FALSE;
            SSMAlterCourse;
         end
      else
         begin
            SSMAlterCourse;
         end;
   end;
```

6.8 END-OF-REPLICATION LOGIC

The end-of-replication logic was denoted `VerifyEOS`. When this procedure is called, it tests whether the missile's range from the FAC(M) is opening or closing. If it is opening, the Boolean variable `EOS` is set to `TRUE` and the two performance measures are incremented. If the SSM-FAC(M) range is less than 30 metres, the missile hit counter is incremented by one. In all end-of-replication events, the miss distance is incremented with the last distance measured when the missile was still closing the target. The Pascal source code to implement this event follows directly.

```
procedure VerifyEOS;
{-----------------------}
{ Check SSM hit criteria }
{-----------------------}
var dist : real;

begin
   dist := sqrt(sqr(SSM.posn[1]-ShipCentre[1])+
                sqr(SSM.posn[2]-ShipCentre[2]));
   if dist > RangeTest then
      begin
         if dist < hitdist then
            begin
               Mhit := Mhit + 1;
               writeln(reportfile,'  HIT      ',RangeTest:10:2);
            end
         else
            begin
               writeln(reportfile,'  MISSED  ',RangeTest:10:2);
            end;
         EOS := TRUE;
         MissDistance := MissDistance + RangeTest;
      end
   else
      begin
         RangeTest := dist;
      end;
end;
```

6.9 THE REPORT FILE

A necessary by-product of the program is the report file named REPORT.TXT. The report file is a text file that contains a list of the replications and their associated miss distances complete with a synopsis of the tactics that were employed, number of hits observed and mean miss distance.

The report file is opened at the beginning of the simulation, updated at the end of every replication in the `VerifyEOS` procedure with the replication number and the associated miss distance. On completion of the simulation, the synopsis of the tactics that were employed, number of hits observed and mean miss distance are added to the file and printed on the screen by the `FinalReport` procedure. A fragment of such a report is in Figure 6.16.

```
REPLICATION :    93  HIT           1.78
REPLICATION :    94  HIT           0.33
REPLICATION :    95  HIT           3.53
REPLICATION :    96  HIT           2.07
REPLICATION :    97  HIT           0.47
REPLICATION :    98  HIT           0.45
REPLICATION :    99  HIT           0.73
REPLICATION :   100  HIT           0.76

TACTICS EMPLOYED

-  Speed remain constant
+  Alter course to maximise SSM angular rate
-  No Jamming
-  Short Range Chaff not deployed
-  Medium Range Chaff not deployed

SIMULATION RESULTS

   Hits Obtained      :   100 out of   100
   Mean Miss Distance :       1.06 meters
```

Figure 6.16: Fragment from a report generated by the simulation model

On completion of the simulation, the report file is closed. Also, note that the opening and closing of the report file is done within the main part of the program whilst the writing of the report is done in the relevant procedures.

6.10 MODEL VALIDATION

At the end of all the relevant chapters, we have indicated why we have assumed that the models described in that chapter are valid or necessary and sufficient. Thus, we have assumed that the models for the environment, FAC(M), SSM and the associated tactics are valid, necessary and sufficient.

It now remains to make an assumption about the validity of the complete model. The logic depicted in Figure 6.1, describes the overall working of the model. We have endeavoured to model the situation at a systems level where we can make conclusive observations about the tactics and their influence on both the missile's hit probability and miss distance. It is submitted that no new intuitions emerged about the level at which the environment, ship, missile and missile counter measures were simulated.

In order to facilitate the building of the model, several assumptions about parameters such as the turning rate and other attributes of the ship and missile were made. As our FAC(M) and SSM are generic concepts, we will accept same as valid. Should inferences about particular missiles or ships be made, the chosen parameters must be updated accordingly. To facilitate this, extensive use was made of data files and constants declared in the program.

All aspects of the program logic, veracity of the various functions and procedures and mathematical correctness were extensively tested and debugged. We are unaware of any remaining problems in regard to the computer model.

Therefore, it is submitted that the overall model and computer version is requisite and, in turn, can be considered valid.

Chapter 7

EXPERIMENTATION AND RESULTS

7.1 INTRODUCTION

The model was computerised by using Turbo Pascal version 7.0. The complete source code is in Appendix G. Furthermore, Appendix E explains the use of all non-standard data types and global variables that were used in the simulation, Appendix F details the inputs and outputs of all functions and procedures used whilst Appendix H contains some programming lessons learnt.

The simulation was run on a Pentium personal computer with a clock speed of 120 MHz and an internal memory of 40 Gb. One simulation of 100 replications took some 50 minutes to complete. Now, if we discard the initialisation of the global variables for the simulation and bearing in mind that we chose a pulse repetition frequency, f, of 500 Hz, the expected run time, t_{run}, of the computer simulation model on a similar computer is simply

$$t_{run} \approx f / 10 \text{ minutes.}$$

Note that we have assumed that there is a linear relationship between simulation run time and pulse repetition frequency. This assumption was made because the simulation model is iterated pulse by pulse and the run time of the generation of discrete events were considered to be negligible.

7.2 MISSILE HIT PROBABILITY

In Section 5.8 we stated that the intention is to minimise the performance measure P(Hit). The outcome of the experiments in regard to the missile's hit probability, are summed up in column (0) in Table 7.1 where the number of hits observed for the 100 replications per simulation is depicted.

Note that the table also shows Yate's Algorithm[1] values for the various columns, that is in all, columns (1) to (5) for the arithmetic calculations, a column for the divisors and a column for the calculated values for the average, main and the interaction effects.

---

[1] E.P. Box, W.G. Hunter and J.S. Hunter, *Statistics for Experimenters*. New York: John Wiley, 1978, pp 342-344.

We denote an entry in column (1) to column (5) of the table in Table 7.1 by e_{rc} where r denotes the row number and c denotes the column number without considering the brackets. Then, for e_{rc} such that $r \in \{1, 2, \ldots, 16\}$ and $c \in \{1,2, \ldots, 5\}$, the entry

$$e_{rc} = e_{2r,c-1} + e_{2r-1,c-1}$$

and for e_{rc} such that $r \in \{17,18, \ldots, 32\}$ and $c \in \{1,2, \ldots, 5\}$, the entry

$$e_{rc} = e_{2(r-16),c-1} - e_{2(r-16)-1,c-1}.$$

A	B	C	D	E	(0)	(1)	(2)	(3)	(4)	(5)	Divisor	Est	ID
-	-	-	-	-	100	200	400	800	823	860	32	26.88	Average
+	-	-	-	-	100	200	400	23	37	-18	16	-1.13	A
-	+	-	-	-	100	200	12	18	-7	-60	16	-3.75	B
+	+	-	-	-	100	200	11	19	-11	18	16	1.13	AB
-	-	+	-	-	100	12	9	0	-23	0	16	0.00	C
+	-	+	-	-	100	0	9	-7	-37	2	16	0.13	AC
-	+	+	-	-	100	11	9	-6	7	0	16	0.00	BC
+	+	+	-	-	100	0	10	-5	11	-2	16	-0.13	ABC
-	-	-	+	-	8	9	0	0	-1	-776	16	-48.50	D
+	-	-	+	-	4	0	0	-23	1	-6	16	-0.38	AD
-	+	-	+	-	0	9	-4	-18	1	-24	16	-1.50	BD
+	+	-	+	-	0	0	-3	-19	1	6	16	0.38	ABD
-	-	+	+	-	7	9	-3	0	1	0	16	0.00	CD
+	-	+	+	-	4	0	-3	7	-1	2	16	0.13	ACD
-	+	+	+	-	0	10	-3	6	-1	0	16	0.00	BCD
+	+	+	+	-	0	0	-2	5	-1	-2	16	-0.13	ABCD
-	-	-	-	+	6	0	0	0	-777	-786	16	-49.13	E
+	-	-	-	+	3	0	0	-1	1	-4	16	-0.25	AE
-	+	-	-	+	0	0	-12	0	-7	-14	16	-0.88	BE
+	+	-	-	+	0	0	-11	1	1	4	16	0.25	ABE
-	-	+	-	+	6	-4	-9	0	-23	2	16	0.13	CE
+	-	+	-	+	3	0	-9	1	-1	0	16	0.00	ACE
-	+	+	-	+	0	-3	-9	0	7	-2	16	-0.13	BCE
+	+	+	-	+	0	0	-10	1	-1	0	16	0.00	ABCE
-	-	-	+	+	6	-3	0	0	-1	778	16	48.63	DE
+	-	-	+	+	3	0	0	1	1	8	16	0.50	ADE
-	+	-	+	+	0	-3	4	0	1	22	16	1.38	BDE
+	+	-	+	+	0	0	3	-1	1	-8	16	-0.50	ABDE
-	-	+	+	+	6	-3	3	0	1	2	16	0.13	CDE
+	-	+	+	+	4	0	3	-1	-1	0	16	0.00	ACDE
-	+	+	+	+	0	-2	3	0	-1	-2	16	-0.13	BCDE
+	+	+	+	+	0	0	2	-1	-1	0	16	0.00	ABCDE

Table 7.1: Number of hits observed, main and interactions effects

Let A^+, B^+, C^+, D^+ and E^+ indicate that the tactics A, B, C, D and E are employed respectively and that A^-, B^-, C^-, D^- and E^- indicate that the respective tactics are not employed. Then we will have

A^+	Increase speed to 30 knots.
B^+	Alter course to maximise the SSM's angular tracking rate.
C^+	Jam the missile head radar to obscure the deployment of close range chaff.
D^+	Deploy close range chaff.
E^+	Deploy medium range chaff.

From Table 7.1, we note that in the first eight experiments, all the missiles have hit their target. The conclusion that we can make from this is that when D^- and E^- are both present, it is immaterial whether A, B and C are at the high or low levels.

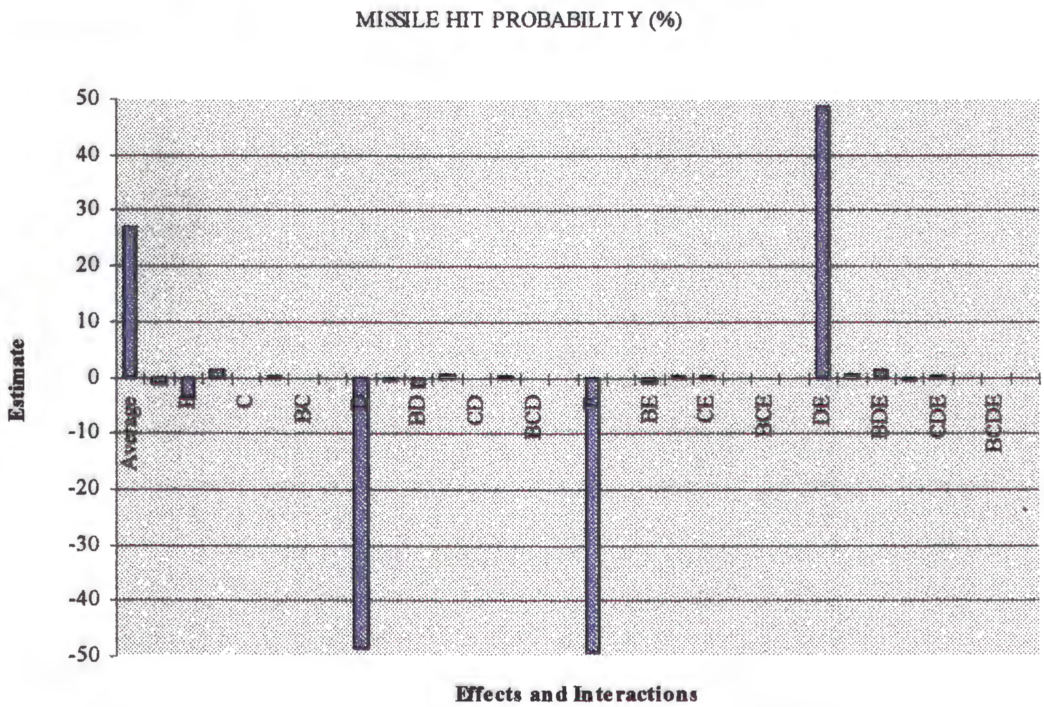


Figure 7.1: Main and interaction effects for missile hit probability[2]

In other words, when chaff is not fired, the other tactics have no effect on the missiles hit probability whatsoever. Also, from our experiments, if we have carried out a 2^3 factorial experiment, considering A, B and C only, the main effects and interaction effects would all be zero.

[2] Note that the sequence of main and interaction effects are exactly as depicted in Table 7.1.

We depict the main and interaction effects graphically in Figure 7.1. We note that, in our 2^5 factorial experiment, the effects of A, B and C are very small in comparison with the effects of D and E. In fact, the main effects associated with A, B and C are significantly different from the main effects associated with D and E at the 1% level.

We note that the effect of C is zero. Thus, jamming the missile head radar has no effect on *P*(Hit) and we will disregard this tactic and its interaction with other tactics as those are in the main also zero or they are not significantly different from zero at the 5% level.

If we consider the experiments where we fire close range chaff and we do not fire medium range chaff (D^+ and E^-), we note that when we maintain our course (B^-), we observed a number of missile hits on the target. However, when we alter course to a course perpendicular to the missile bearing (B^+) we do not observe any missile hits on the target.

Furthermore, if we maintain our speed and course (A^- and B^-), the number of hits when we select to fire close range chaff (D^+) is approximately double to when we increase speed and maintain course (A^+ and B^-). Thus, we see that, in terms of missile hit probability, the effect of the interaction AD is small and negative (-0.38). However, our intention is to minimise *P*(Hit) and therefore, we will not consider this interaction more fully.

If we consider the firing of medium range chaff (E^+) and disregard whether we have fired close range chaff (D^+ or D^-), a picture similar to the scenario in the previous paragraph emerges. However, we note that the effect of the interaction DE is large and positive. Thus, we cannot make a statement about tactics D and E in isolation.

Thus, at this stage conclude that, in order to minimise *P*(Hit), the best missile counter measure is either to

- alter course perpendicular to the missile bearing and to fire close range chaff; or

- alter course perpendicular to the missile bearing and fire medium range chaff.

From the above, we note that we do not have conclusive evidence to point us at one particular combination of tactics that will give us the best result. All we know, is that both the options above will minimise the hit probability of the missile or *P*(Hit).

7.3 MISSILE MISS DISTANCE

The second performance measure is missile miss distance. In Section 5.8 we stated that we want to maximise the missile mean miss distance

$$d_m = \frac{1}{n}\sum_{i=1}^{n} x_i \,.$$

Table 7.2 tabulates these results in a similar manner to what Table 7.1 did for missile hit probability.

A	B	C	D	E	(0)	(1)	(2)	(3)	(4)	(5)	Divisor	Est	ID
-	-	-	-	-	1.77	3.62	6.00	12.00	3176.40	11302.7	32	353.21	Average
+	-	-	-	-	1.85	2.38	6.00	3164.40	8126.38	1018.88	16	63.68	A
-	+	-	-	-	1.06	3.60	1580.47	4065.64	330.90	1513.26	16	94.58	B
+	+	-	-	-	1.32	2.40	1583.93	4060.74	687.98	261.04	16	16.32	AB
-	-	+	-	-	1.77	615.49	2032.87	0.68	694.66	-1.64	16	-0.10	C
+	-	+	-	-	1.83	964.98	2032.77	330.22	818.60	2.14	16	0.13	AC
-	+	+	-	-	1.06	618.16	2032.87	342.50	89.44	4.64	16	0.29	BC
+	+	+	-	-	1.34	965.77	2027.87	345.48	171.60	-7.18	16	-0.45	ABC
-	-	-	+	-	277.36	914.92	0.34	-2.44	3.46	3147.50	16	196.72	D
+	-	-	+	-	338.13	1117.95	0.34	697.10	-5.10	332.52	16	20.78	AD
-	+	-	+	-	430.15	914.85	165.45	406.10	-0.68	705.94	16	44.12	BD
+	+	-	+	-	534.83	1117.92	164.77	412.50	2.82	80.00	16	5.00	ABD
-	-	+	+	-	279.17	914.92	171.29	0.40	-1.84	-1.44	16	-0.09	CD
+	-	+	+	-	338.99	1117.95	171.21	89.04	6.48	2.30	16	0.14	ACD
-	+	+	+	-	430.41	909.20	171.29	90.12	1.26	4.48	16	0.28	BCD
+	+	+	+	-	535.36	1118.67	174.19	81.48	-8.44	-7.46	16	-0.47	ABCD
-	-	-	-	+	425.89	0.08	-1.24	0.00	3152.40	4949.98	16	309.37	E
+	-	-	-	+	489.03	0.26	-1.20	3.46	-4.90	357.08	16	22.32	AE
-	+	-	-	+	504.90	0.06	349.49	-0.10	329.54	123.94	16	7.75	BE
+	+	-	-	+	613.05	0.28	347.61	-5.00	2.98	82.16	16	5.13	ABE
-	-	+	-	+	425.90	60.77	203.03	0.00	699.54	-8.56	16	-0.53	CE
+	-	+	-	+	488.95	104.68	203.07	-0.68	6.40	3.50	16	0.22	ACE
-	+	+	-	+	504.88	59.82	203.03	-0.08	88.64	8.32	16	0.52	BCE
+	+	+	-	+	613.04	104.95	209.47	2.90	-8.64	-9.70	16	-0.61	ABCE
-	-	-	+	+	425.89	63.14	0.18	0.04	3.46	-3157.3	16	-197.33	DE
+	-	-	+	+	489.03	108.15	0.22	-1.88	-4.90	-326.56	16	-20.41	ADE
-	+	-	+	+	504.90	63.05	43.91	0.04	-0.68	-693.14	16	-43.32	BDE
+	+	-	+	+	613.05	108.16	45.13	6.44	2.98	-97.28	16	-6.08	ABDE
-	-	+	+	+	420.17	63.14	45.01	0.04	-1.92	-8.36	16	-0.52	CDE
+	-	+	+	+	489.03	108.15	45.11	1.22	6.40	3.66	16	0.23	ACDE
-	+	+	+	+	506.67	68.86	45.01	0.10	1.18	8.32	16	0.52	BCDE
+	+	+	+	+	612.00	105.33	36.47	-8.54	-8.64	-9.82	16	-0.61	ABCDE

Table 7.2: d_m observed, main and interactions effects

Also, Figure 7.2 depicts the situation in Table 7.2 graphically.

MISS DISTANCE

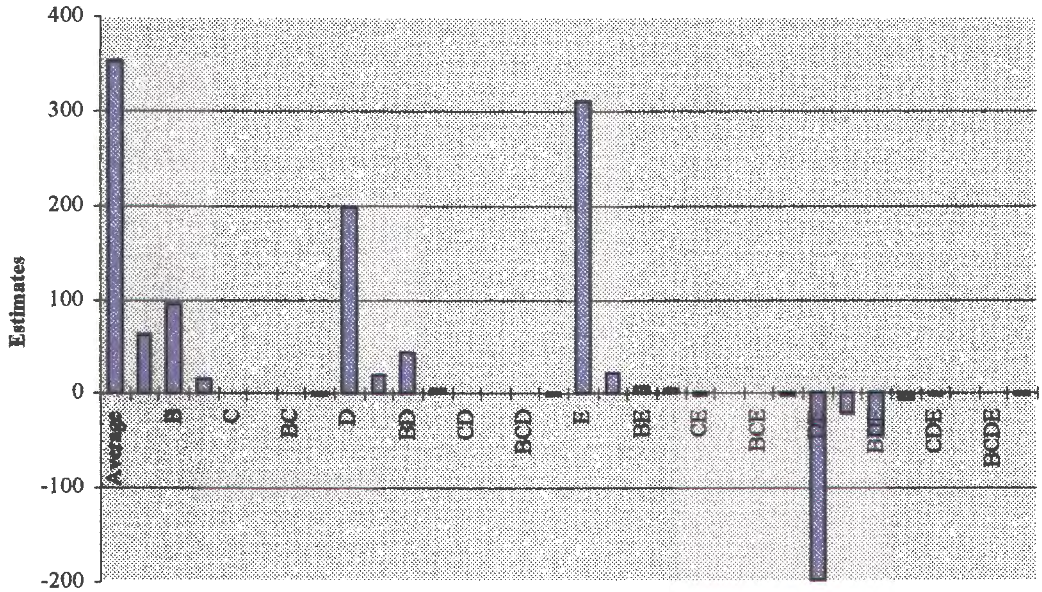


Figure 7.2: Main and interaction effects for miss distance[3]

Again, we note by inspection of Table 7.2 and Figure 7.2 that the main effect of jamming (C) as well as the effects of the interactions of C with the other tactics tends to zero. However, in Section 3.5.4 we assumed that

$$P(\text{Operator see SSM}|\ \text{Radar detects SSM}) = 1.$$

Thus, bearing this assumption in mind, we assume that jamming has no noteworthy influence on any of the other tactics and as a result, we disregard jamming and its interactions with the other tactics. However, we reserve our opinion about jamming in cases where

$$P(\text{Operator see SSM}|\ \text{Radar detects SSM}) < 1.$$

In regard to tactics A and B, we see that their main effects are more pronounced than in the case of the performance measure $P(\text{Hit})$. However, the effects of their interactions with one another and with tactics D and E are relatively small. We conclude by saying that we accept that increasing speed to 30 knots and changing course perpendicular to the missile bearing has a positive influence on the miss distance.

In regard to tactics D and E, again we see major main effects with a very large interaction between D and E. This time, as expected, the main effects are positive whilst the effect of the interaction DE is large and negative. Therefore, we again have the situation where it is difficult to decide between tactics D and E. However, from

[3] Note that the sequence of main and interaction effects are exactly as depicted in Table 7.2.

Table 7.2, we note that the estimated main effect of tactic D is 196.72 metres whilst the estimated main effect of tactic E is 309.37 metres. This represents an approximate ratio of 2:3 between tactic D and tactic E. Now, if we consider tactics A and B together with tactic D as tested by experiment E11010 and tactics A and B together with tactic E as tested by experiment E11001, we test to see whether their mean values are significantly different by means of a t-test performed by the SYSTAT software, we find that they are significantly different at a level less than 0.1% (See Figure 7.3).

```
SYSTAT VERSION 5.0
COPYRIGHT, 1990-1994
SYSTAT, INC.

 PAIRED SAMPLES T-TEST ON E11001 VS E11010 WITH 100 CASES

 MEAN DIFFERENCE =        78.219
 SD DIFFERENCE   =        62.972
 T =       12.421 DF =    99 PROB =         0.000
```

Figure 7.3: Paired samples t-test on E11001 versus E11010

Also, we note from Table 7.3 that the mean miss distance for experiment E11001, denoted m_{E11001}, was 613.05 metres whilst the mean miss distance for experiment E11010, denoted m_{E11010}, was 534.83 metres. Thus we conclude that $m_{E11001} > m_{E11010}$ at a level of less than 0.1%.

However, By inspection of Table 7.3, we note that experiment E11011 also produced a mean miss distance, $m_{E11011} = 613.05$. Again we endeavour to test whether there is a significant difference in the means m_{E11001} and m_{E11011}. However, we note that $m_{E11001} = m_{E11011}$ and that their standard deviations $s_{E11001} = s_{E11011} = 165.36$. Therefore, we can deduce that there is no significant difference between m_{E11001} and m_{E11011}.

Furthermore, if we calculate the Pearson moment correlation coefficient, ρ, between experiment E11001 and experiment E11011, we find that $\rho = 1$. Therefore, we assume that the cause of the individual miss distances in every replication was due to the effects of tactic E, because, in every replication, the value of the miss distance for both experiment E11001 and experiment E11011 was the same. We submit that, although ρ shows correlation only and is not an indicator of cause and effect, this evidence, together with the main effects of tactics D and E, the interaction effects of interactions ABD, ABE and DE, and the result of our t-test above, suggests no other explanation.

Again, if our assumption that $P(\text{Operator see SSM}| \text{Radar detects SSM}) = 1$ holds, we assume that the optimum combination of tactics to counter an oncoming SSM is to increase speed to 30 knots, alter course perpendicular to the missile bearing and to fire medium range chaff.

7.4 CONCLUSION

Our aim was to show how FAC(M) missile counter measures against a SSM can be optimised by simulating a generic ship and generic missile in a particular environment. We submit that the aim was achieved. However, we must bear in mind that the simulation technique is a heuristic method where the simulation serves to quantify how well the chosen tactics will perform in terms of the chosen performance measures (See Section 8.5).

Furthermore, many of our assumptions might be critical. An example is the assumption that P(Operator see SSM| Radar detects SSM) = 1. Suppose that

$$P(\text{Operator see SSM}|\ \text{Radar detects SSM}) = 0.25$$

and that the ESM receiver is also only 25% effective. This would mean that the missile will invariably be locked onto its target before the operator or other systems will detect the missile. In this case, we can argue that tactic C may be of some value, tactic E might have no effect and that tactic D interacting with tactic C might be a better solution. In other words, by jamming the missile head radar to degrade the range tracking loop, time might be gained to deploy close range chaff and offer alternative targets to the missile in close proximity of the ship, which, in turn, might save the day. Under such circumstances, medium range chaff might not work at all as the missile range gate is positioned inside the medium range chaff pattern. We shall discuss this matter in more detail in the next chapter.

Chapter 8

ASSUMPTIONS AND RECOMMENDATIONS

8.1 INTRODUCTION

In this final chapter we will revisit some of the assumptions that we have made in order to build the model and make recommendations as to their applicability in a South African naval context, their general validity, future research to be carried out and suggested areas of refinement in the model. The assumptions and their applicable recommendations will be dealt with under the following headings:

- The environment.
- The FAC(M) model.
- The SSM model.
- Tactics to be evaluated.
- The Simulation model.

8.2 THE ENVIRONMENT

8.2.1 AREA OF OPERATIONS

We have chosen an area of operations that spans the Southern African seaboard from the equator to 45°S and from the Greenwich meridian to 50°E. During the years 1970 to 1994, this could have been an appropriate demarcation of the area of operations as the South African government of that time was engaged in military operations in adjoining countries to the north. Also, the choice of the area was convenient as the South African Maritime Data Centre for Oceanology data base covers that particular area. However, since May 1994, the political situation in South Africa has changed radically.

The present government's military policy is of a defensive nature[1]. Also, recent events such as the South African initiative to broker the transition of Zaïre into the Democratic Republic of the Congo would indicate that this government is ready to

[1] Republic of South Africa, Parliament, *White Paper on Defence*, 1996, p6.

play a major role in African diplomatic affairs. Furthermore, Von Clausewitz[2] holds that war is a mere continuation of policy by other means. We conclude that, for the South African government, to play a diplomatic role in Africa, it must back up that role by its military.

Also, if one considers the formation of the Southern African Development Community (SADC) and its military sub-structure, the Interstate Defence and Security Committee (ISDSC), it points to a formation of aligned African states in the sub-continent. Also, the security of these states would depend on relations with states bordering or nearly bordering on the SADC countries. As a result, we assume that the South African government is likely to extend its diplomatic role in Africa to the whole of sub-Saharan Africa.

Therefore, we recommend that in future research, the area of operations to be extended to at least 12°N.

8.2.2 THE SIMULATION AREA OF OPERATIONS

We chose the simulation area of operations as the worst area in the area of operations in regard to weather experienced. This approach led to statistically strict estimates of the sea and wind conditions for the simulation model. However, it can be argued that such an approach may bias the outcome of the model in that the effectiveness of the tactics under consideration is not tested for milder weather conditions.

For that end, it is recommended that the simulation is also conducted with wind, current and temperature parameters that reflect the general conditions in the whole of the area of operations.

8.3 THE FAC(M) MODEL

8.3.1 OVERALL CONCEPT

At present, the South African Navy's surface warfare capability is centred around nine WARRIOR-class FAC(M). However, this situation is not likely to endure into the future. Some of these ships have been in commission in the South African Navy since 1977[3]. This represents a period of two decades. Considering the pace of modern technological advancement, the ships and their associated equipment can be considered outdated. Great Britain, France and other countries recently made proposals to the South African government for the purchase of corvettes or small guided missile armed frigates (NATO designation: FSG). FSGs normally displaces between 1 800 and 3 000 tons.

From a simulation point of view, FSGs will have a major impact on our model. Radar cross section, turning rates and acceleration data, to name a few, will be significantly different. In the event that should the South African Navy acquire such ships, it is

---

[2] Carl Von Clausewitz, *On War*. Edited with an introduction and notes by Anatol Rapoport. Harmondsworth, England : Penguin Books. [1832] 1968. p. 119.

[3] *Jane's Fighting Ships 1995/96*, edited by Capt R. Sharpe RN , London: Jane's Yearbooks, [1995]

submitted that it would be prudent to optimise missile counter tactics for these ships. Although this dissertation shows how missile counter measures can be optimised for FAC(M), we submit that the methodology developed in this dissertation is necessary and sufficient in order to do likewise for FSGs. Our motivation is that the model is generic. Thus it is simply a matter of substituting values to allow the optimisation of missile counter measures for FSGs. It is recommended that this methodology is used to optimise missile counter measures for future acquisitions of the South African Navy.

8.3.2 RADAR CROSS SECTION

In order to simulate the radar cross section of a FAC(M), we have assumed that such a ship is already protected by some stealth measures such as the application of radar absorbent material. This led to reduced values for the radar cross section simulation parameters. Also, we have assumed expected values for radar cross section simulation. We have accepted the use of expected values for radar cross section because we argued that the automatic gain control of a missile would result in such values. However, it can also be argued that the statistical nature of radar cross section should be modelled because the automatic gain control response might not be sufficient to ensure expected radar cross section values.

Therefore, it is recommended that the statistical nature of radar cross section should be modelled and simulated to ascertain whether or to what degree the model is sensitive to that aspect.

8.3.3 MANOEUVRING CHARACTERISTICS

The manoeuvring characteristics of our model was validated by a trial conducted by SAS RENÉ SETHREN. The result of that trial confirmed that the estimation of turning rates and the acceleration constant in this dissertation is sufficient. However, it is recommended that a sensitivity analysis be carried out to ascertain to what extent the model is sensitive to those parameters.

The choice of a Weibull distribution to model reaction time for the commencement of changes in speed and course should be further investigated by sufficiently observing typical reaction times in order to verify the veracity of the assumption.

8.3.4 RADAR DETECTION OF AN ONCOMING MISSILE

We have made the assumption that if the oncoming missile is detected by radar, the radar operator will observe the fact, or

$$P(\text{Operator see SSM}|\text{Radar detect SSM}) = 1.$$

As a result, one can argue that the model is biased towards missiles being detected at extended ranges or, in other words, the failure of an operator to observe an oncoming missile at maximum detection range has not been incorporated in the model.

Furthermore, according to blip-scan theory[4], an operator will only observe the first blip that appears on the cathode ray tube and will decide that it is a legitimate radar echo only after several blips have been observed. Also, the probability that a blip will paint on the cathode ray tube if a target is in the radar beam, is not necessarily one. See Figures 3.11 and 3.12 for detail on this probability. It is recommended that these factors be taken into account in future iterations of the model.

8.3.5 ELECTRONIC WARFARE EQUIPMENT

The electronic support measures were deemed to detect the oncoming missile at range R'_{ESM} provided the missile head radar was activated when

$$R'_{ESM} = \min\left(R_{ESM} \sim \text{normal}\left(R_{SR}, R_{SR}/2\right), R_{HORIZON}\right) \tag{8.1}$$

and where R_{ESM} is the theoretical detection range of the electronic support measures equipment, R_{SR} is the radar detection range and $R_{HORIZON}$ is the calculated radar horizon. It is recommended that a similar argument to the argument against the radar detection model can be made. It is recommended that this model also be subjected to sensitivity analysis in order to establish to what extent the model is sensitive to (8.1) .

The use of a repeater jammer such as a range gate stealer was omitted from the model on the assumption that it will not be effective against missile head radar with leading edge tracking. It is submitted that the assumption be tested by including a range gate stealer in a subsequent iteration of our model.

The chaff models modelled chaff as three discrete points, each with their own expected radar cross section. This approach is considered adequate. However, we have seen that radar cross section is statistical in nature[5] and as in Section 8.3.2, it is recommended that the statistical nature of radar cross section should be modelled and simulated to ascertain whether or to what degree the model is sensitive to that aspect. Furthermore, the placement of both close range and medium range chaff should be varied to ascertain how sensitive the model is to the initial positions of the chaff bundles.

8.4 THE SSM MODEL

8.4.1 HOMING METHODS

SSM homing methods in regard to percentage of SSM makes with particular homing methods were depicted in Figure 4.1. We chose the radar homing method as the only homing method for our generic missile as this represented 72% of missile makes worldwide. The next largest category is radar homing with infra-red alternatives (15%) and the third largest category was depicted as a combination of radar and infra-red homing (8%). If the objective of the simulation is to find whether a particular set of tactics would be effective against an infra-red tracker, it is submitted that our

---

[4] *Naval Operations Analysis*. 2 ed. Annapolis, Ma: Naval Institute Press. 1977. pp 91-93.

[5] L. B. Van Brunt, *Applied ECM - Vol 1*. Dunn Loring, Va.: EW Engineering, c1978. pp 379-381.

simulation model will have to be revised substantively to enable such research. However, often an infra-red tracking device is complimentary to the missile head radar. By this we mean that the infra-red device ascertains whether a target being tracked by the missile head radar has an infra-red signature. This information is used to verify that the missile head radar is tracking a valid target. If not, the missile head radar will discard the target and commence a search for a valid target. A possible tactic against such a missile would be to deploy infra-red candles with close and medium range chaff. Furthermore, such missiles could operate in both or either of the medium or long wave infra-red spectra[6], that is the spectra associated with 3 to 6 μm and with 6 to 15 μm wave lengths respectively.

The spectral characteristics of all material can be classified as blackbody emitters, greybody emitters and selective radiators. Whereas the blackbody emitter can be considered as a theoretical emitter, the greybody emitter approximates natural occurring objects such as ships' hulls, personnel, terrestrial and space objects. These normally emit energy in both the medium and long wave infra-red spectra. For our purpose, the most important selective radiator is the ship's exhaust gasses. These radiators can be found in the medium wave infra-red spectrum.

It is recommended that, in future development, the simulation should include the use of infra-red target verification in both the medium and long wave spectra.

8.4.2 NUMBER OF MISSILES FIRED

In Chapter 4, we have assumed that a single missile will be fired at our generic FAC(M) by an opposing force. Also, we assumed that, in line with modern missile development, such a missile could be pre-programmed to fly through any way-point to obscure the opposing force's position.

It is submitted that an opposing force may consider firing more than one missile at the FAC(M). Furthermore, the opposing force might also decide to program their missiles to approach the target from different directions. Such actions will have the effect of complicating the FAC(M)'s missile counter measures.

It is recommended that the effect of firing more than one SSM at a FAC(M) be considered a high priority for future development of the model.

8.4.3 SPEED

We chose our generic missile speed to be Mach 0.9 in order to accommodate the majority of SSMs. However, from Table B-2, we note that China, France and Russia have developed SSMs that are capable of speeds in excess of Mach 1. Also, the international SSM, designated Hellfire, is purported to cruise at a speed which is in excess of Mach 1.

These missiles are still in the minority. However, they represent a trend in modern missile warfare that cannot be ignored. For ships to survive such missiles, short

[6] W.H. Gunter, *Naval Infrared Handbook - Volume 1*. Simon's Town: IMT, 1996. p 88.

reaction times will be crucial. Deployment of chaff and other tactics will have to be carried out much faster. For example, a missile cruising at Mach 1.8 and detected at 15 kilometres, will reach its target in about 25 seconds as opposed to our generic missile that will take about 50 seconds to reach its target under the same conditions. Also, because the technology to fly missiles at speeds in excess of Mach 1 at low altitudes are still not perfected, we can assume that such missiles will have very different flight profiles. We may find that, in order to simulate these missiles, we will have to amend our model substantially.

8.4.4 MISSILE HEAD RADAR

In Section 4.3.3 we made a series of assumptions about the missile head radar. The major considerations were wavelength, peak power, pulse length, pulse repetition frequency, antenna gain and beam width as well as certain aspects of discrimination and general errors. From these assumptions, we have predicted expected detection range. It is submitted that the detection range of a radar could be sensitive to variations in the assumed parameters. For example, the Justice radar simulation indicates that if a 3 cm wavelength is chosen, the detection range where $P(\text{Detect}) = 1$, increases by roughly 10% in comparison to our chosen missile head radar parameters.

If the objective is to study a particular missile, the above-mentioned parameters should be determined as exactly as possible.

8.4.5 SIMULATION OF THE RANGE GATE

In simulating the range gate, we have considered the radar reflective energy in the range gate as discrete points with a particular radar cross section. Also, we have chosen a relatively sparse radar cross section model for our generic FAC(M). After extensive experimentation with the algorithm that was developed in Section 4.4, we concluded that the algorithm was sufficient for our purpose. However, in order to implement leading edge tracking, the algorithm was found lacking. This can be attributed to the fact that the radar cross section model for the ship can be considered sparse. In order to rectify the situation, one of the following two approaches can be used.

Firstly, a more dense radar cross section model for the FAC(M) can be implemented. Such an approach could be considered relatively difficult. In Section 3.3, we have explained that although mathematical models to find the radar cross section of an object exists, they only hold for very simple shapes. The radar cross section can be measured, but, to allocate such a measurement to many discrete points may not be feasible.

Secondly, by using a limited number of discrete points such as we have done, is relatively easy to set up. If one considers the length of the radar pulse and incorporate same in the algorithm, one should be able to overcome the problem of a sparse radar cross section model. For example, if the pulse length of a particular tracking radar represents ten meters in distance, a non-sparse representation of our model can be achieved. See Figure 8.1 for the situation where three discrete points, complete with the simulated radar returns taking a pulse length of ten metres into account, are depicted. Note that the red area depicts the energy returning to the missile head radar from the particular radar reflective point 1 for a pulse length of ten metres, etcetera. Furthermore, the height of the shaded area depicts the summed signal strength at any particular point.

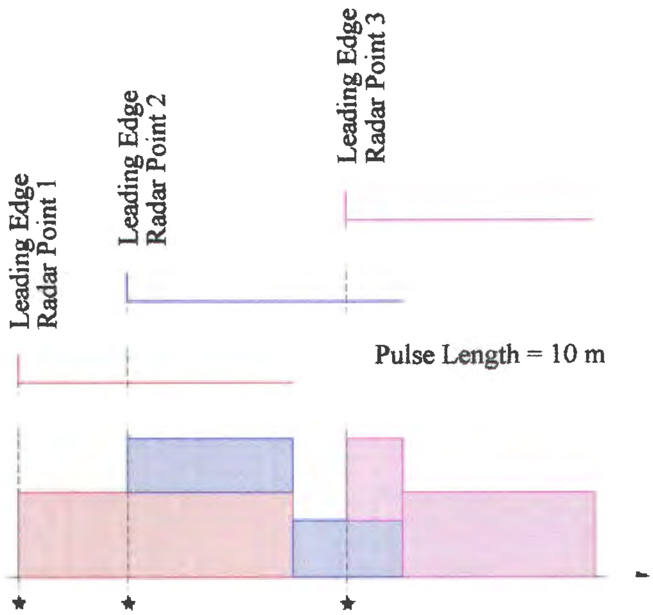


Figure 8.1: Radar cross section points extended by pulse length

It is recommended that the latter method be used to enable leading edge tracking.

8.4.6 BALLISTIC ALGORITHM

We assumed that the missile will behave similarly to a ballistic projectile in the last 150 metres of its flight. This was done to stabilise the missile's flight path in the final stages by avoiding violent course changes. However, one can make an assumption that the ballistic phase is software driven, that is, some computer instruction disallows the execution of further directional commands. It then follows that, if the missile's fuse is not activated within a prescribed time, the missile may carry out some manoeuvre and search pattern to endeavour to find another target. It is recommended that this option be investigated.

8.4.7 PROPORTIONAL NAVIGATION

In Section 4.9, we stated that proportional navigation can be achieved by turning the velocity vector of the missile at a rate proportional to the line-of-sight rate. We defined this turning rate in (4.12). Also, we modified (4.12) in (4.15) so as to be able to calculate a lead angle for the missile. However, not all proportional navigation algorithms are implemented in this fashion. Jordan[7] and Zarchan[8] both propose that

[7] G.M. Jordan, *Class Notes for a Short Course in Homing Guided Missiles*, Ann Arbor, Mi: FAAC. c1975, p 4.
[8] P. Zarchan, *Tactical and Strategic Missile Guidance*, 2 ed., Washington: AIAA, c1994.p 26.

the missile should turn at the rate expressed in (4.12) until $\dot{\lambda} = 0$. Thereafter the missile must continue at the course it was flying at the time when the value of $\dot{\lambda}$ became zero.

If this method is chosen, it will mean that the missile must be able to change course at various rates as ordered by the proportional navigation algorithm. It is not known if such a method will improve our model and it is recommended that the model's sensitivity in regard to various methods for implementing proportional navigation be investigated.

8.4.8 AERODYNAMICS

Our algorithm for change the missile's course is equivalent to regarding the changing course of the missile as a first order control system. Also, it assumes a fixed rate of course change, that is, 70°/s. In order to implement the alternative proportional navigation methods described in the previous section, our missile will have to be able to change course at various rates. We submit that this can be easily implemented. Furthermore, we have disregarded the possibility to model the missile's aerodynamic characteristics by a second order control system, or a differential equation of the form

$$\frac{d^2y}{dt^2} + 2\zeta\omega_n\frac{dy}{dt} + \omega_n^2 y = \omega_n^2 x$$

where the constant ζ is the damping ratio and ω_n is the undamped natural frequency of the missiles turning characteristics.

It is recommended that it must be ascertained to what degree our model is sensitive to these various approaches.

8.4.9 HIT CRITERIA

We have assumed that our generic missile has three methods of activating its detonating chain, viz.,

- direct action;
- deceleration; and
- proximity fuse.

However, in practice, this might not be the case. Therefore, it is recommended that variations of the three methods be tried to establish whether they have any influence on the simulated outcomes. It is submitted that, by inspection of the data, one can draw a conclusion that the model might be insensitive to the second and third methods as the miss distances measured where, in general, less than the half-width of the FAC(M).

8.5 TACTICS TO BE EVALUATED

Whilst developing the model, we noted that there are five general areas that lend themselves to the development of tactics. These are changes in speed, changes in course, jamming, the deployment of close range chaff and the deployment of medium range chaff. In our model, we have chosen one example in every area. However, we know that simulation as an optimisation tool can be regarded as a heuristic[9]. Thus, optimal results will only be obtained if we can optimise the tactics to be evaluated ourselves. The simulation only quantifies the expected value of the tactic under consideration.

In order to see the effect of changing speed, we chose a tactic whereby speed is increased to 30 knots. However, we may even consider decreasing speed or stopping altogether. Thus there is many speed tactics that can be considered.

Similarly, there are many changing course tactics to choose from. Two additional tactics that might be worthwhile investigating are to

- alter course to minimise radar cross section; and to
- alter course to open all gun arcs.

As far as jamming is concerned, we might not only want to use a spot jammer to obscure the firing of close range chaff after lock on by the missile, but we may want to use a repeater jammer to generate alternative targets for the missile head radar to track initially.

The patterns used for close range chaff can also be varied. For example, we may choose, under certain conditions, to fire close range chaff astern of the ship. Furthermore, the effect of rotating launchers to place chaff as far away as possible, but still in the missile head radar's range gate should also be investigated.

The patterns fired for medium range chaff are also of importance. For example, we can decide to place the medium range chaff further away whilst increasing the number of rockets fired. This might have the effect of increasing the expected miss distance. Also, by not firing any medium range chaff rockets directly at the target, we may reduce the expected hit probability of the missile when we maintain course and speed.

The above discussion is aimed at showing the vast array of tactics and combinations thereof that is possible. However, to make such a comprehensive study, would be nonsensical if real life ship and missile parameters are not used. By the unclassified nature of this dissertation, this was not feasible. However, the student of naval warfare would be wise to undertake such a study.

[9] G.N. Engelbrecht, *A Guide to Numeric Simulation Techniques*. Simon's Town: SA Navy, 1996. p 12.

8.6 THE SIMULATION MODEL

The model was programmed for computer execution in Turbo Pascal version 7.0, using a structured programming paradigm. Although this was sufficient for our purposes, it is submitted that, for the following three reasons, an object-orientated paradigm would be more suitable:

- The implementation of many varying tactics could be accomplished much easier.

- The generation of more than one missile would not require much more source code as the missiles would inherit their characteristics from their class.

- The ability to think about the objects in a computerised system as the corresponding physical objects in a real life system is a considerable aid in the production of understandable systems[10].

Furthermore, the program was written in such a way that is must be used within its integrated development environment. This was done to minimise development time. However, for anyone else using the program, he must be familiar with the Pascal language and the Borland integrated development environment.

Also in the development of the computer model, sixty constants, or variables with constant values, were defined. These constants represent almost all the values that appertain to our assumptions. Thus, the sensitivity of many of our assumptions can be tested by changing the values associated with the constants.

In terms of the computer model, we make the following recommendations:

- The program should be rewritten using an object-orientated paradigm.

- The program should be adapted to run outside an integrated development environment to facilitate use by users that are not capable of programming.

- Users should be able to vary the assumptions and their associated values by a simple interface.

---

[10] D. Bell, I. Morrey and J. Pugh. *Software Engineering - A Programming Approach*. London: Prentice Hall, 1987. p 124.

Appendix A

ABBREVIATIONS

AA	:	Anti-Aircraft
AGC	:	Automatic Gain Control
CPA	:	Closest point of approach.
CRC	:	Close Range Chaff
DCT	:	Depth Charge Thrower
ECM	:	Electronic Counter Measure
Elint	:	Electronic Intelligence
ESM	:	Electronic Support Measure
EW	:	Electronic Warfare
FAC(M)	:	Fact Attack Craft armed with Missiles
FSG	:	Small Frigate armed with Missiles
HMS	:	Her Majesty's Ship (UK)
HOJ	:	Home-on-Jam
INS	:	Israeli Naval Ship
IP	:	Initial Position
IR	:	Infra-Red
LET	:	Leading Edge Tracking
LOS	:	Line-Of-Sight
MER	:	Maximum Effective Range
MHR	:	Missile Head Radar
MRC	:	Medium Range Chaff
MTI	:	Moving Target Indication
NATO	:	North Atlantic Treaty Organisation
nm	:	Nautical Mile(s)
OPFOR	:	Opposing Force
RCS	:	Radar Cross Section
RGS	:	Range Gate Stealer
Rx	:	Receiver
SAN	:	South African Navy
SANDF	:	South African National Defence Force
SAS	:	South African Ship
SS	:	Submarine
SSM	:	Surface-to-Surface Missile
SU	:	Surface
TAS	:	Torpedo and Anti-Submarine
Tx	:	Transmitter

Appendix B

TACTICAL SHIPBORNE SURFACE-TO-SURFACE MISSILES (1996) [1]

Abbreviation	Meaning
α	Alternative Method
C	Command or Beam Ride
D	Diameter
H	Home or Fire and Forget
I	Inertial
IR	Infra-Red
L	Length
λ	Laser Semi-Active
M	Mid-Course Guidance
R	Radar
U	Unknown Method
W	Wire Guided

Table B.1 : LEGEND

Country	**Dimensions (cm)**			**Weight**	**Speed**	**Range (km)**		**Missile Control**				
Designation	L	D	Wing	(Kg)	(Mach)	Min	Max	R	IR	H	C	W
China												
C-101	650	54	162	1850	2.0	-	24	✓		✓		
C-201	736	76	240	1700	0.9	-	80	✓		✓		
FL-1	642	76	2400	2300	0.9	-	54	✓	α	✓		
FL-2	600	54	170	1550	0.9	-	50	U	U			
FL-7	660	54	170	1800	1.4	-	30	U	U			
HY-1	580	76	240	2300	0.9	-	21	✓	α	✓		
HY-2	736	76	240	2500	0.9	-	40	✓	α	✓		
SY-1	580	76	240	2300	0.9	-	21	✓	α	✓		
YJ-1	581	36	118	815	0.9	4.5	23	✓		✓		
YJ-2	639	36	118	715	0.9	14.5	65	✓		✓		
France												
ANNG	538	38	96	860	2.0	-	80	✓		✓		
MM 15	230	18	56	103	0.9	1.5	7	✓			✓	
MM 38	521	35	100	735	0.9	2	22	✓		✓		
MM 39	521	35	100	660	0.9	2	27	✓		✓		
MM 40 B 1	578	35	113	855	0.9	2	38	✓		✓		
MM 40 B 2	580	35	113	870	0.9	2	41	✓		✓		
SS 12M	186	21	65	76	0.6	-	3					✓
International												
Hellfire	162	18	36	48	1.0+	-	5					λ
Otomat Mk1	466	46	136	762	-	-	86	✓		✓		
Otomat Mk2	446	46	136	770	-	-	86	✓		✓		
Israel												
Gabriel Mk 1	335	34	135	430	0.7	-	11				✓	
Gabriel Mk 2	342	34	135	522	0.7	-	19				✓	

[1] *Jane's Naval Weapon Systems*, edited by E.R. Hooten, 1996, London : Jane's Yearbooks , 1996, SSM Section.

Country	Dimensions (cm)			Weight	Speed	Range (km)		Missile Control				
Designation	L	D	Wing	(Kg)	(Mach)	Min	Max	R	IR	H	C	W
Italy												
Sea Killer 2	470	21	98	300	0.9	-	14	✓		✓		
Japan												
SSM-1B	508	34	-	660	0.9	-	80	✓		✓		
Norway												
Penguin 1	300	28	142	330	0.7	1.5	10		✓			
Penguin 2	300	28	142	340	0.8	1.5	14		✓			
Russia												
SSN-2A	580	76	240	2300	0.9	3	24	✓		✓		
SSN-2B	580	76	240	2300	0.9	3	43	✓		✓		
SSN-2C	655	76	240	2500	0.9	3	46	✓	✓	✓		
SSN-2D	655	76	240	2600	0.9	3	54	✓	✓	✓		
SSN-3B	1020	98	370	4600	1.2	12	190	✓	α	✓		
SSN-7	650	76	U	2700	0.9	-	60	✓	α	✓		
SSN-12	1170	88	210	4600	1.7	-	296	✓		✓	M	
SSN-19	1000	85	U	3250	1.6	-	296	✓		✓	M	
SSN-22	938	130	U	3950	2.0	-	48				I	
SSN-25	378	42	93	480	0.9	2.5	70	✓		✓		
Sweden												
RBS 15M	435	50	1.4	780	0.8	-	38	?	?	✓		
Taiwan												
Hsiung Fen I	390	34	135	540	0.9	-	19				✓	
Hsiung Fen II	465	39	135	685	0.85	-	70	✓	✓	✓		
United Kingdom												
Sea Skua SL	250	25	72	145	0.85	-	8				✓	
United States of America												
Harpoon 1A	463	34	83	519	0.85	-	50	✓		✓	I	
Harpoon 1B	463	34	83	519	0.85	-	50	✓		✓	I	
Harpoon 1C	463	34	83	519	0.85	-	67	✓		✓	I	
Tomahawk A/B/C/D	SS Fired Nucleur SSMs with ranges up to 1350 nm											

Table B-2 : Tactical Shipborne Surface-to-Surface Missiles (1996)

Appendix C

THE FAC(M) FLEET (1996) [1]

Country	*Class*	*Number*	*Displacement (Tons)*	*Associated SSM*
Algeria	OSA I	2	171	SSN-2A
	OSA II	9	210	SSN-2B
Azerbaijan	OSA II	1	210	SSN-2B
Bahrain	Al Manama	2	632	Exocet MM 38
	Ahmed el Fatah	4	228	Exocet MM 40
Bangladesh	Huangfen	4	171	Hai-Ying
	Hegu	4	79	SY-1
Brunei	Waspada	3	206	Exocet MM 38
Bulgaria	OSA I	1	171	SSN-2B
	OSA II	3	210	SSN-2B
Cameroon	P 48S	1	308	Exocet MM 40
Chile	SAAR 3	2	250	Gabriel II
	SAAR 4	2	415	Gabriel II
China	Houijian	3	520	YJ-1
	Houxin	11	480	YJ-1
	OSA I	100	171	YJ-1
	Hegu	75	68	SY-1
Croatia	Koncar	1	242	SSN-2B
	OSA I	1	171	SSN-2B
Cuba	OSA I	5	171	SSN-2
	OSA II	13	210	SSN-2
Denmark	Flyvefisken	14	450	Harpoon
	Willemoes	10	260	Harpoon
Ecuador	Manta	3	228	Exocet MM 38
Egypt	Ramadan	6	307	Otomat Mk 1
	OSA I	6	171	SSN-2A
	October	6	82	Otomat Mk 1
	Hegu	6	68	SY-1
	Komar	2	79	SSN-2A
Ethiopia and Eritrea	OSA II	1	245	SSN-2A
Finland	Helsinki	4	300	Saab RBS 15
	Rauma	4	248	Saab RBS 15SF
	Tuima	2	245	SSN-2B
Gabon	FAC(M)	1	160	SS 12M
Germany	Gepard	10	391	Exocet MM 38
	Albatros	10	398	Exocet MM 38
	Tiger	16	265	Exocet MM 38

[1] *Jane's Fighting Ships 1995/96*, edited by Capt R. Sharpe , London : Jane's Yearbooks, [1995].

Country	*Class*	*Number*	*Displacement (Tons)*	*Associated SSM*
Greece	Combattante III Combattante IIA Combattante II	10 4 4	425 265 255	Exocet MM 38 Exocet MM 38 Exocet MM 38
India	OSA II	8	245	SSN-2A/B
Indonesia	Dagger	4	270	Exocet MM 38
Iran	Combattante II Hegu OSA II	10 10 1	275 68 245	Harpoon or YJ-1 YJ-1 SSN-2B
Iraq	OSA I	1	171	SSN-2A
Israel	SAAR 4.5 SAAR 4	5 8	488 450	Harpoon Harpoon/Gabriel II
Italy	Sparvierro (Hydrofoil)	6	61	Otomat Mk 2
Ivory Coast	Patra	2	147	SS 12 M
Kenya	Nyayo	2	400	Otomat Mk 2
Korea (North)	Soju OSA I Huangfen Komar Sohung	15 8 4 10 10	220 171 171 85 85	SSN-2 SSN-2A SSN-2A SSN-2A SSN-2S
Korea (South)	Pae Ku Ashville Wildcat	8 1 2	268 245 140	GDC/Harpoon GDC Exocet MM 38
Kuwait	TNC 45 FPB 57	1 1	255 410	Exocet MM 40 Exocet MM 40
Latvia	Storm OSA I	1 5	135 171	Unknown Ex Norway Unknown Ex GDR
Libya	Combattante II OSA II Susa	9 12 3	311 245 95	Otomat Mk 2 SSN-2C SS 12 M
Lithuania	Storm OSA I	1 3	135 171	Unknown Ex Norway Unknown Ex GDR
Malaysia	Spica Perdana	4 4	240 234	Exocet MM 38 Exocet MM 40
Mexico	Isla Coronado	4	52	MM 15
Morocco	Lazaga	4	425	Exocet MM 38
Nigeria	Lürsen 57 Combattante III	3 3	444 430	Otomat Mk 1 Exocet MM 38
Norway	Hauk Storm Snögg	14 10 6	148 135 135	Penguin Mk 2 Penguin Mk 1 Penguin Mk 1
Oman	Dhofar	4	394	Exocet MM 40
Pakistan	Huangfen Hegu	4 4	171 68	Hai Ying 2 SY-1
Peru	Velarde	6	470	Exocet MM 38
Poland	OSA I	7	171	SSN-2A
Qatar	Vita Combattante III	4 3	376 395	Exocet MM 40 Exocet MM 40
Romania	OSA I	6	171	SSN-2
Russia and associated states	Tarantul Matka OSA II OSA I	49 11 10 4	455 260 245 171	SSN-2C SSN-2C SSN-2B SSN-2A
Saudi Arabia	Al Siddiq	9	478	Harpoon

Country	*Class*	*Number*	*Displacement (Tons)*	*Associated SSM*
Singapore	Victory	6	550	Harpoon
	Sea Wolf	6	254	Harpoon/Gabriel II
South Africa	Minister[2]	9	430	Skerpioen
Sweden	Hugin	12	170	Penguin Mk 2
	Norrköping	12	230	Saab RBS 15
Syria	OSA I	2	171	SSN-2A
	OSA II	10	245	SSN-2C
	Komar	5	85	SSN-2A
Taiwan	Lung Chiang	2	250	Hsiung Feng I
	Hai Ou	50	47	Hsiung Feng I
Thailand	Ratcharit	3	270	Exocet MM 38
	Prabparapak	3	268	Gabriel I
Tunisia	Combattante III	3	425	Exocet MM 40
Turkey	Yildiz	5	436	Harpoon
	Dogan	8	436	Harpoon
	Kartal	8	190	Penguin Mk 2
United Arab Emirates	Mubarraz	2	260	Exocet MM 40
	Ban Yas	6	260	Exocet MM 40
United States of America	Pegasus (Hydrofoil)	6	240	Harpoon
Venezuela	Constitución	6	170	Otomat Mk 2
Vietnam	OSA II	8	245	SSN-2B
Yemen	Tarantul	2	580	SSN-2C
	OSA II	5	245	SSN-2B
Yugoslavia[3]	OSA I	6	171	SSN-2A
	Koncar	3	242	SSN-2B

Table C-1 : List of FAC(M) deployed worldwide (1996)

[2] Renamed WARRIOR-class on 1 April 1997

[3] Present situation unknown

Appendix D

RESULTS OF RADAR DETECTION SIMULATION

D.1 SEARCH RADAR DETECTION OF AN ONCOMING MISSILE

The graphs below depict the probability that a missile will be detected by a search radar in a given sea state. The Justice Radar Simulation was developed by the Institute for Maritime Technology, Simon's Town, and it was used with their kind permission. Note that the sea state is expressed in the Beaufort scale.

D.1.1 SIMULATION CONSTANTS

The simulation uses the following constants:

Radar Type	: Search Radar : F-band.
Transmitter	
Wavelength	: 10 cm.
Peak Power	: 450 kW.
Pulse Length	: 4 μs.
PRF	: 500 hz.
Tx Loss	: 2 dB.
Receiver	
Noise Level	: 6 dB.
System Losses	: 9 dB.
Rx Losses	: 5 dB.
Pulses Integrated	: 32.
Integration	: Coherent.
Antenna	
Gain	: 34 dB.
Vertical Beam Width	: 8.3°.
Horizontal Beam Width	: 1.2°.
Scan Rate	: 12 r.p.m.
Side Lobe Level	: -30 dB.
Height	: 17 m.
Polarisation	: Vertical.
Discrimination	
Type	: Signal Processing.
Processing Losses	: 2 dB.

Errors

Imp Height Error	: 3.5 m.
Timing Jitter	: 15 ns.
Height S/N Error	: 10 m.
Bearing S/N Error	: 10 mr.
Elevation S/N Error	: 10 mr.

D.1.2 RADAR STATISTICS FOR VARIOUS SEA STATES

The detection probabilities of an oncoming missile with a RCS of 1 m^2 proceeding at 310 m/s for various sea states are given in the table below.

Sea State	Detection Probability (Beyond Sea Horizon)				
	0.00	0.25	0.50	0.75	1.00
0	32.68	29.97	28.96	27.81	22.21
1	32.50	30.14	29.12	27.81	22.33
2	32.29	29.97	28.78	27.96	22.21
3	32.29	29.97	28.96	27.81	22.33
4	31.93	29.97	28.96	27.81	22.47
5	32.29	29.97	29.12	27.96	22.33
6	32.12	29.97	28.96	27.81	22.33
7	32.50	29.97	28.96	27.81	22.47

Table D-1: Cumulative Detection Probability for various sea states and ranges in kilometres.

D.1.3 SIMULATION OUTPUTS

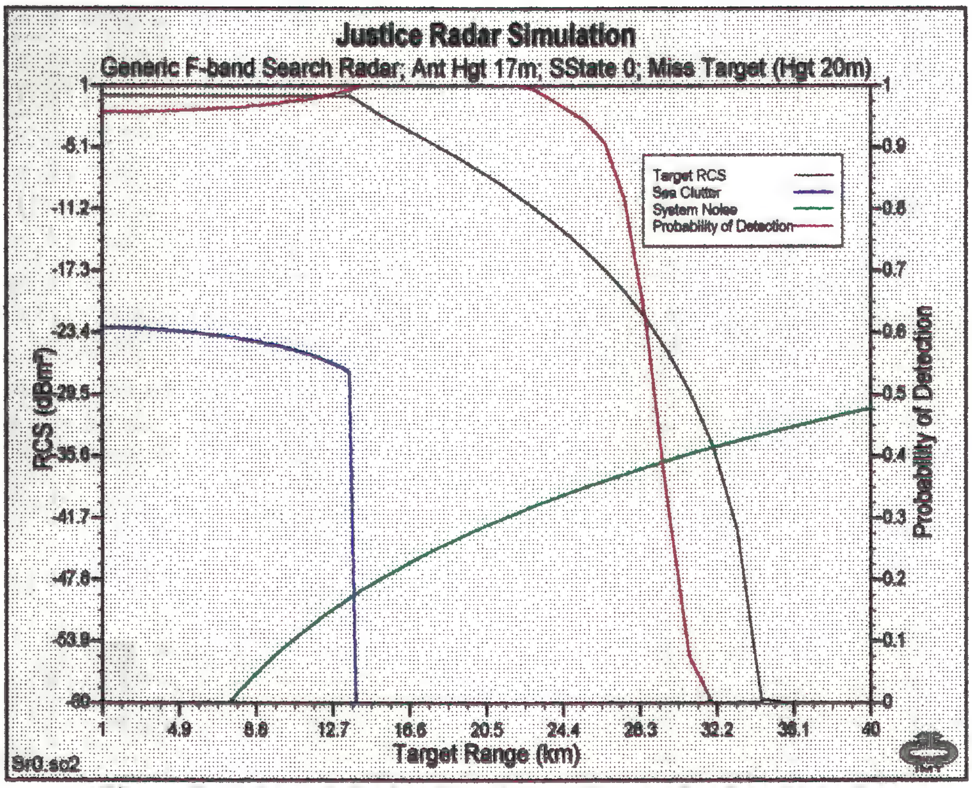


Figure D.1: Search Radar Simulation Output for Sea State 0.

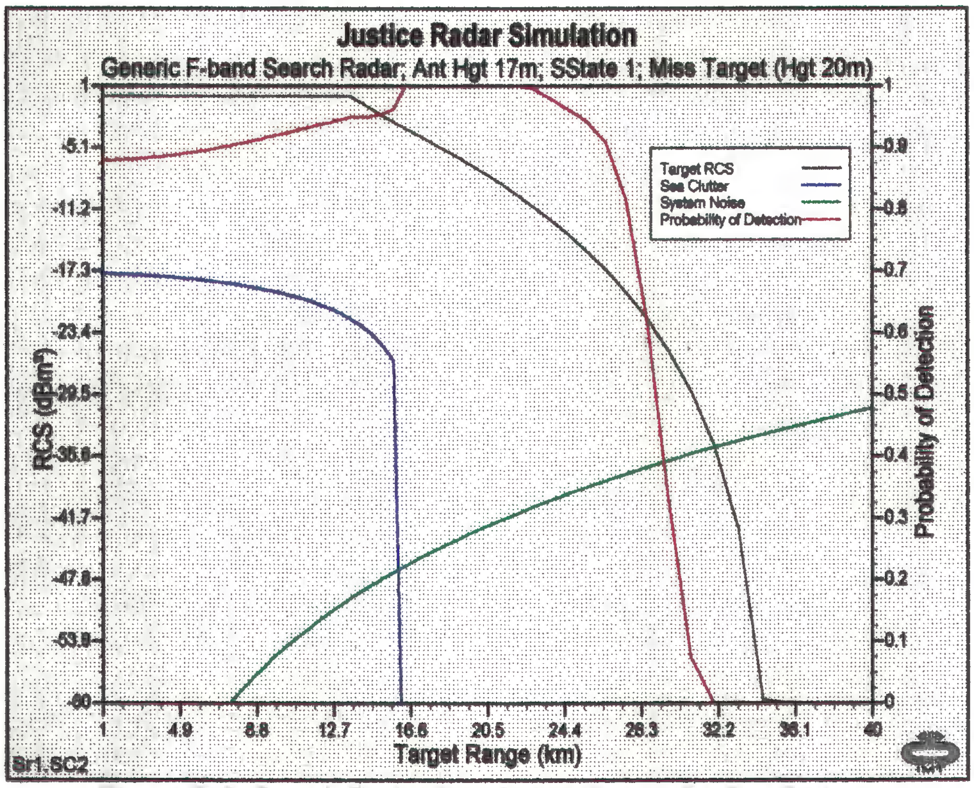


Figure D.2: Search Radar Simulation Output for Sea State 1.

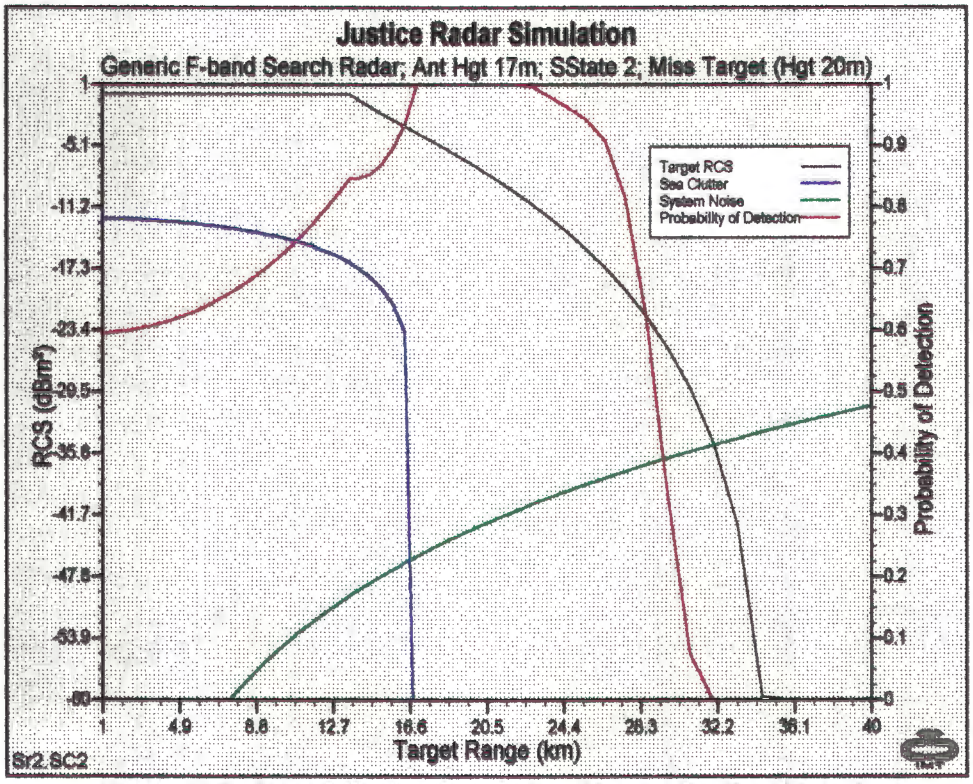


Figure D.3: Search Radar Simulation Output for Sea State 2.

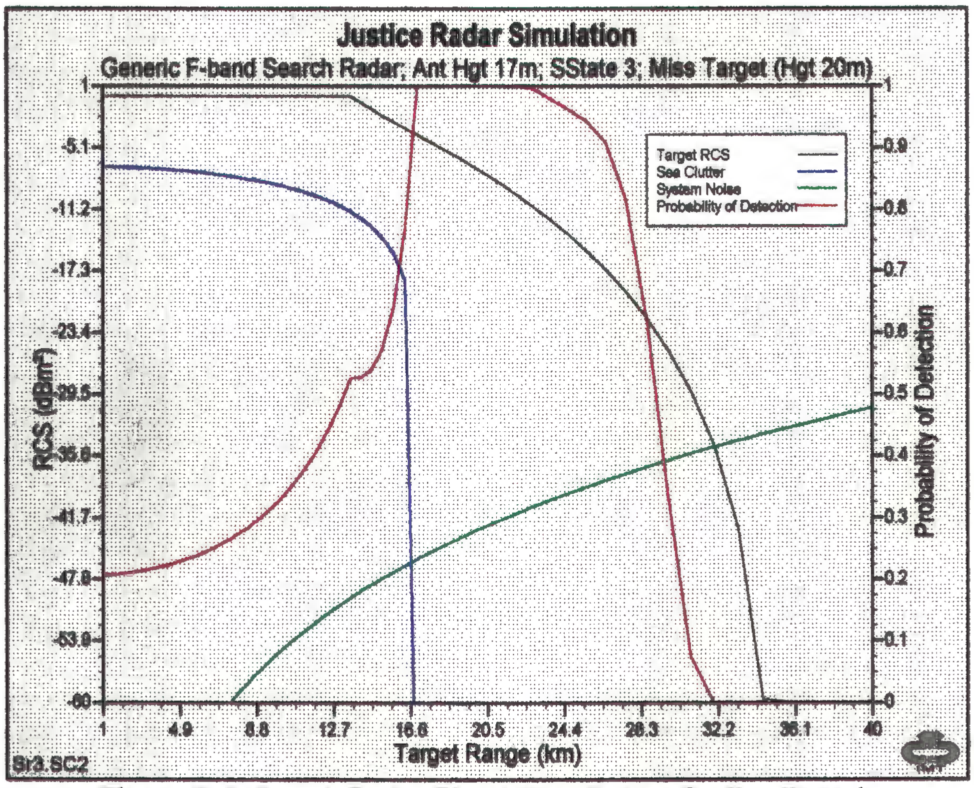


Figure D.4: Search Radar Simulation Output for Sea State 3.

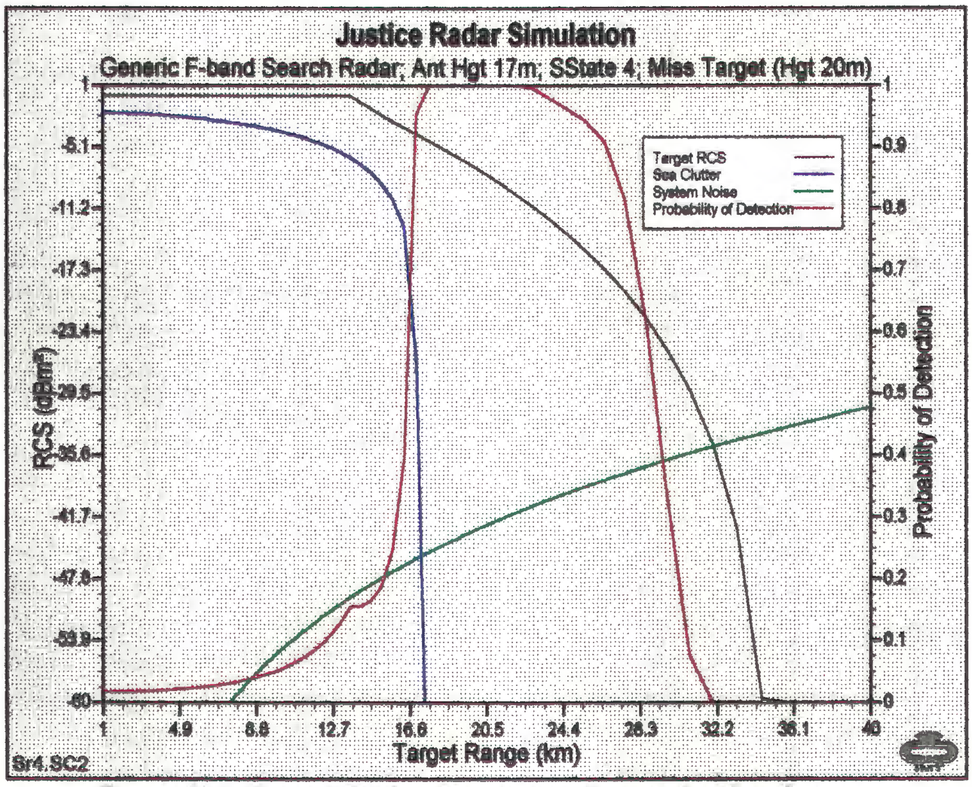


Figure D.5: Search Radar Simulation Output for Sea State 4.

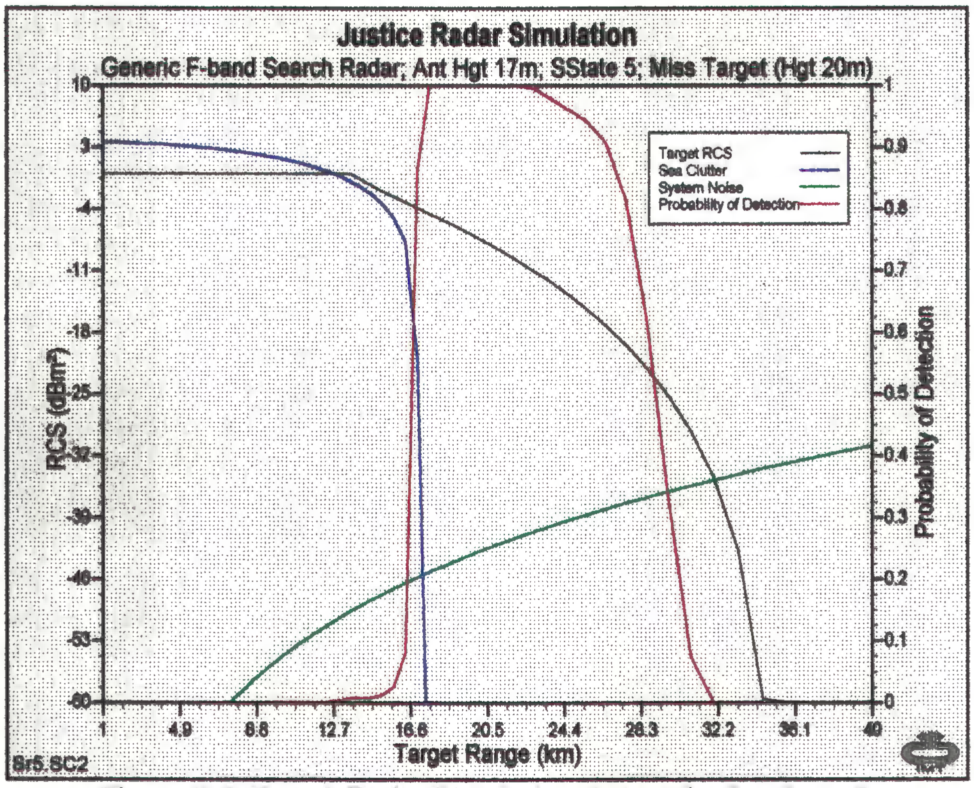


Figure D.6: Search Radar Simulation Output for Sea State 5.

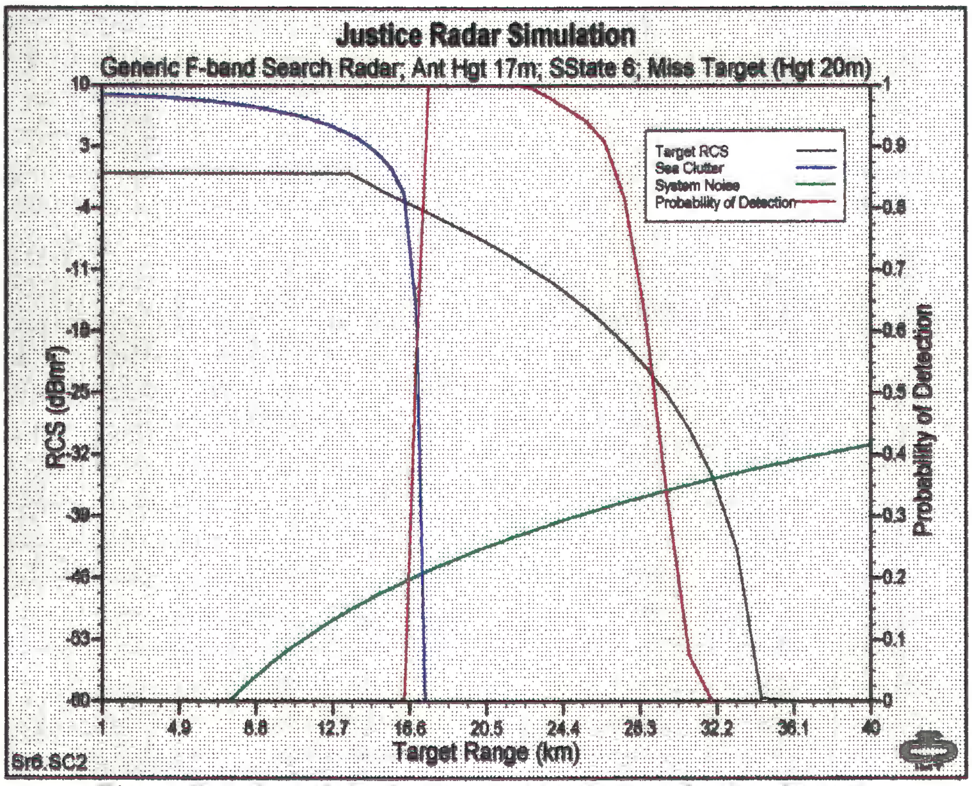


Figure D.7: Search Radar Simulation Output for Sea State 6.

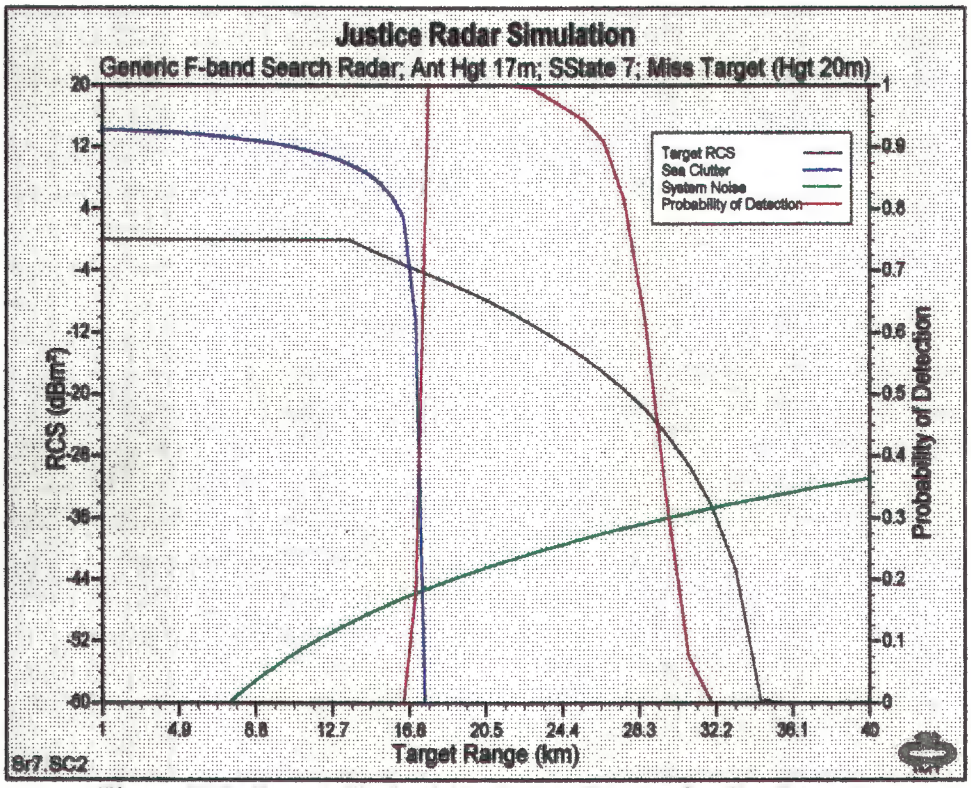


Figure D.8: Search Radar Simulation Output for Sea State 7.

D.2 MISSILE HEAD RADAR DETECTION OF A FAC(M)

The graphs below depict the probability that a ship will be detected by a MHR in a given sea state. However, we will first give the radar constants used for the simulation.

D.2.1 SIMULATION CONSTANTS

The simulation uses the following constants:

Radar Type	: Tracking Radar (K-band).
Transmitter	
Wavelength	: 1 cm.
Peak Power	: 20 kW.
Pulse Length	: 1 μs.
PRF	: 500 hz.
Tx Loss	: 1 dB.
Receiver	
Noise Level	: 7 dB.
System Losses	: 9 dB.
Rx Losses	: 6 dB.
Pulses Integrated	: 10.
Integration	: Coherent.

Antenna

Gain	: 28 dB.
Vertical Beam Width	: 7°.
Horizontal Beam Width	: 2°.
Side Lobe Level	: -15 dB.
Height	: 20 m.
Polarisation	: Vertical.

Discrimination

Type	: Signal Processing.
Processing Losses	: 2 dB.

Errors

Imp Height Error	: 3 m.
Timing Jitter	: 9 ns.

Note that the chosen parameters are fictitious in the sense that they do not represent a particular SSM, nor do they reflect any classified information. However, the constants were chosen such that they are realistic in regard to modern radar technology. At least two radar and/or missile experts have indicated that it is indeed the case[1].

D.2.2 SIMULATION OUTPUTS

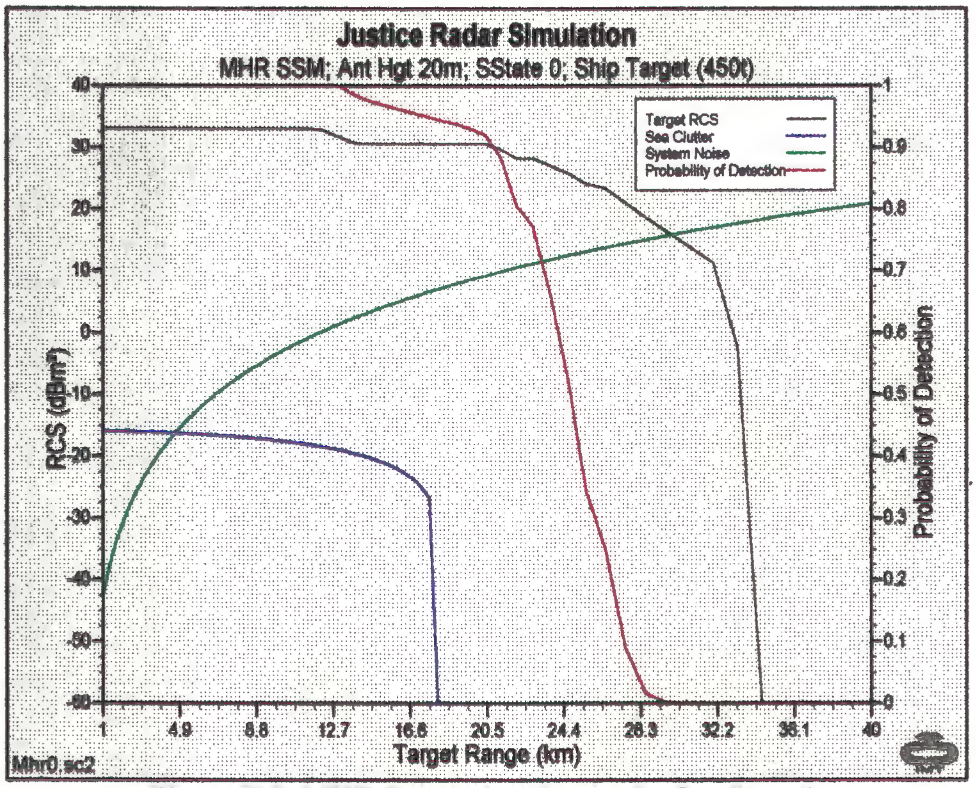


Figure D.9: MHR Simulation Output for Sea State 0.

[1] P. Botha, Radar Researcher, IMT, Simon's Town. Personal Interview, 23 April 1997, Simon's Town. Åke Svensson., Project Manager, Missile Division, Saab Dynamics AB, Linköping, Sweden. Personal Interview, 22 April 1997. Pretoria.

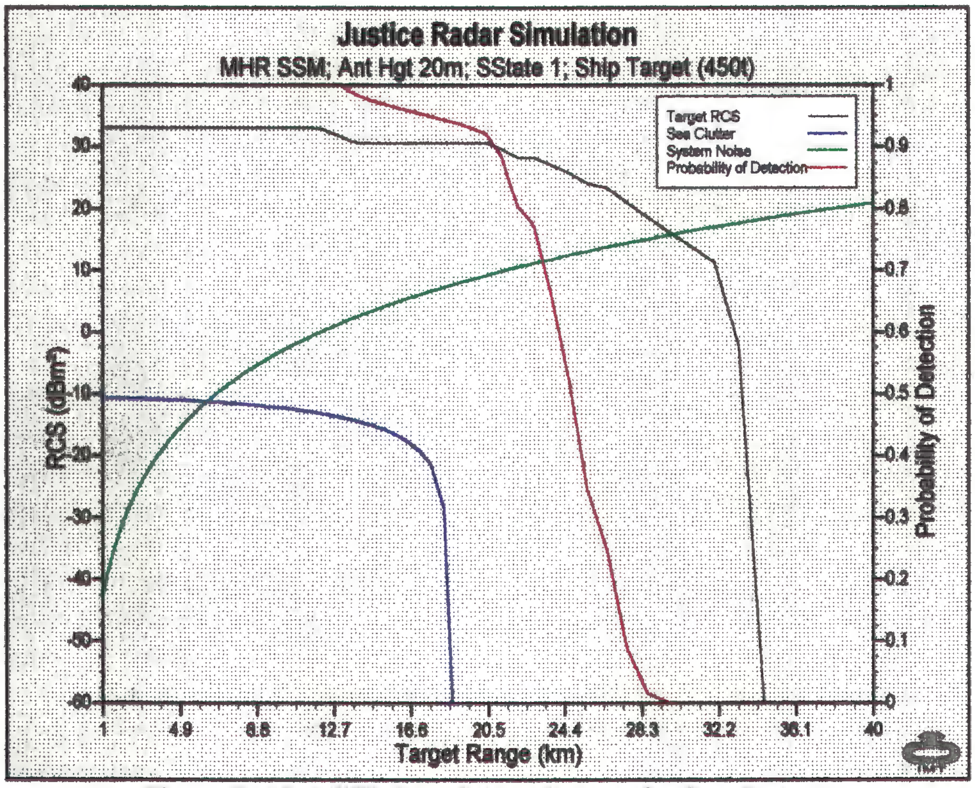


Figure D.10: MHR Simulation Output for Sea State 1.

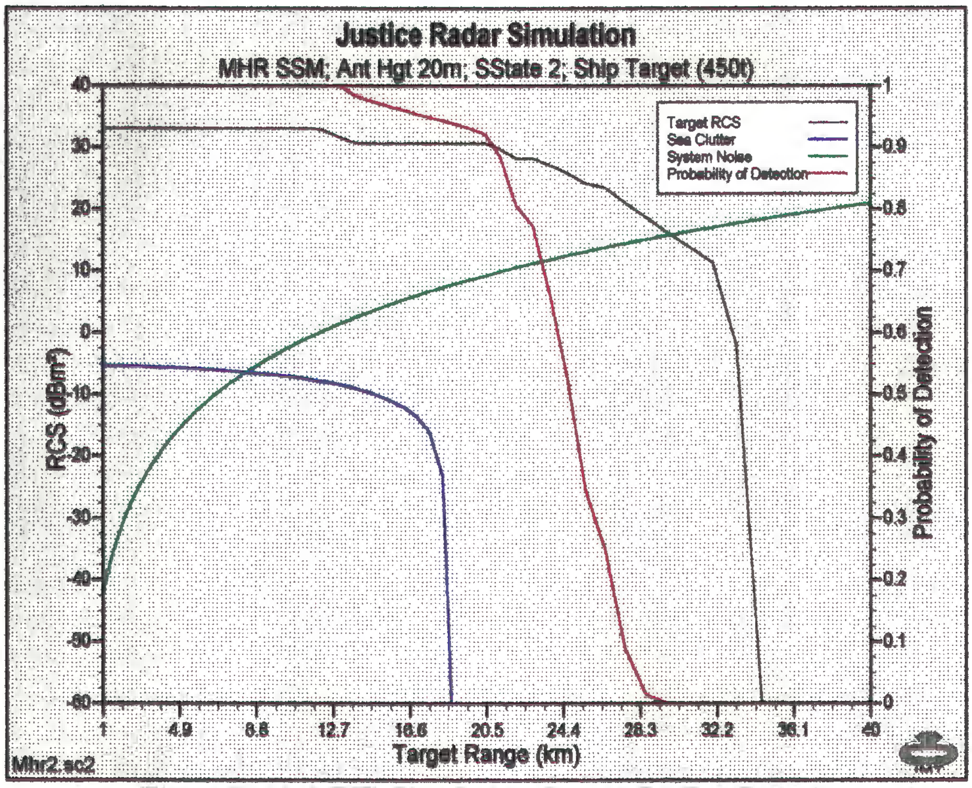


Figure D.11: MHR Simulation Output for Sea State 2.

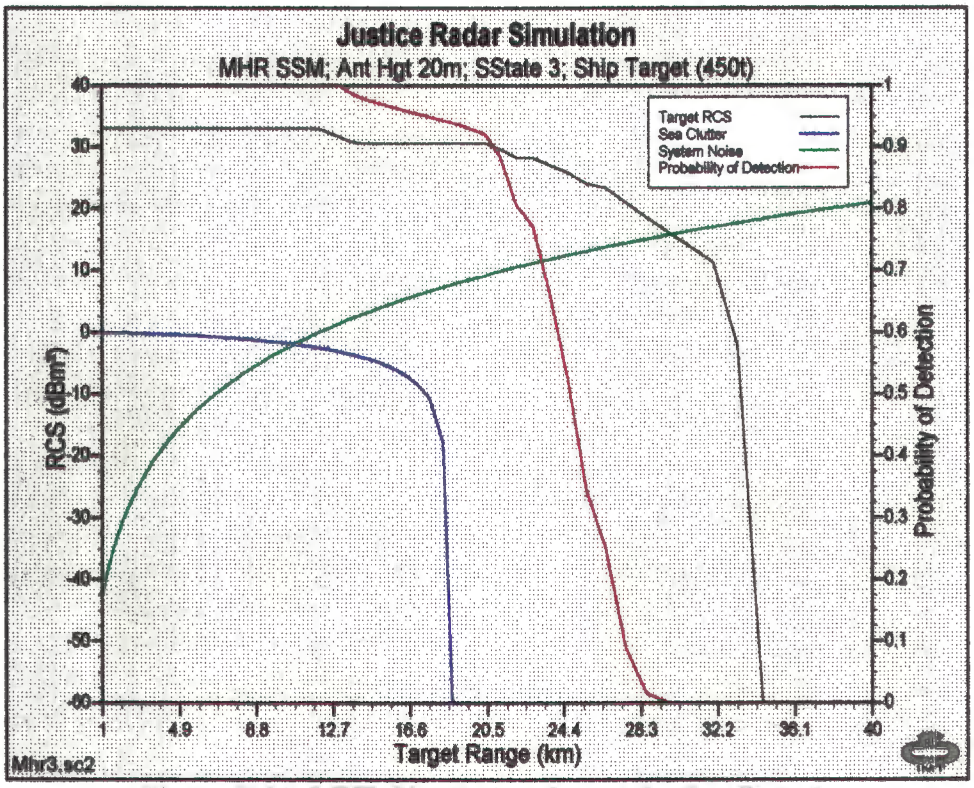


Figure D.12: MHR Simulation Output for Sea State 3.

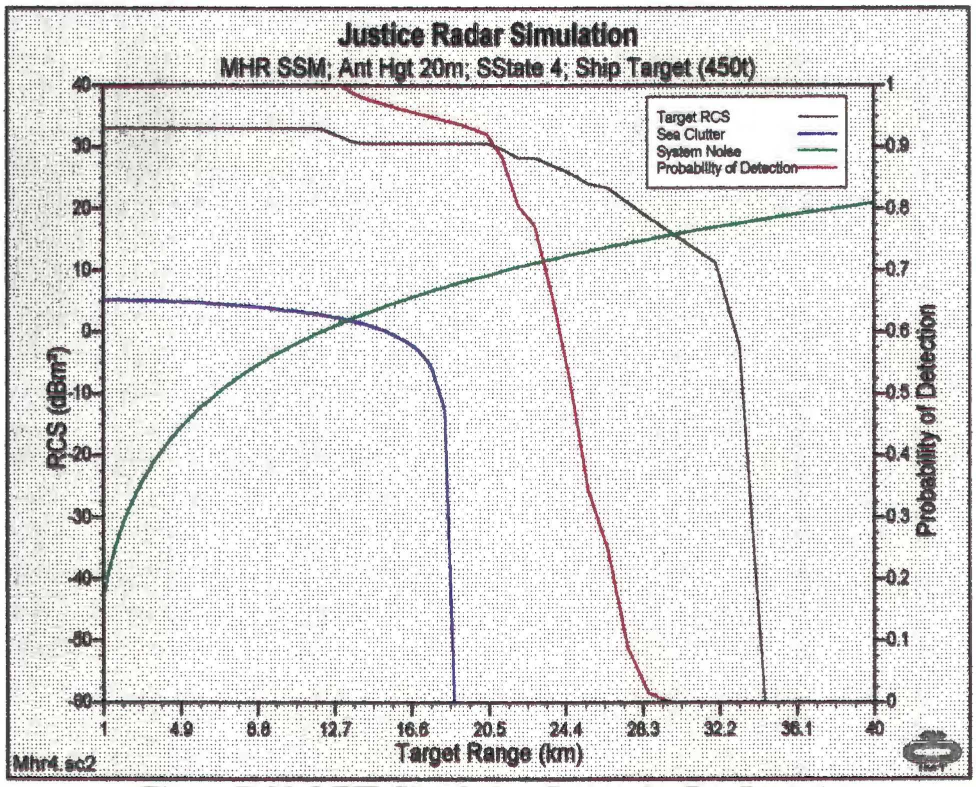


Figure D.13: MHR Simulation Output for Sea State 4.

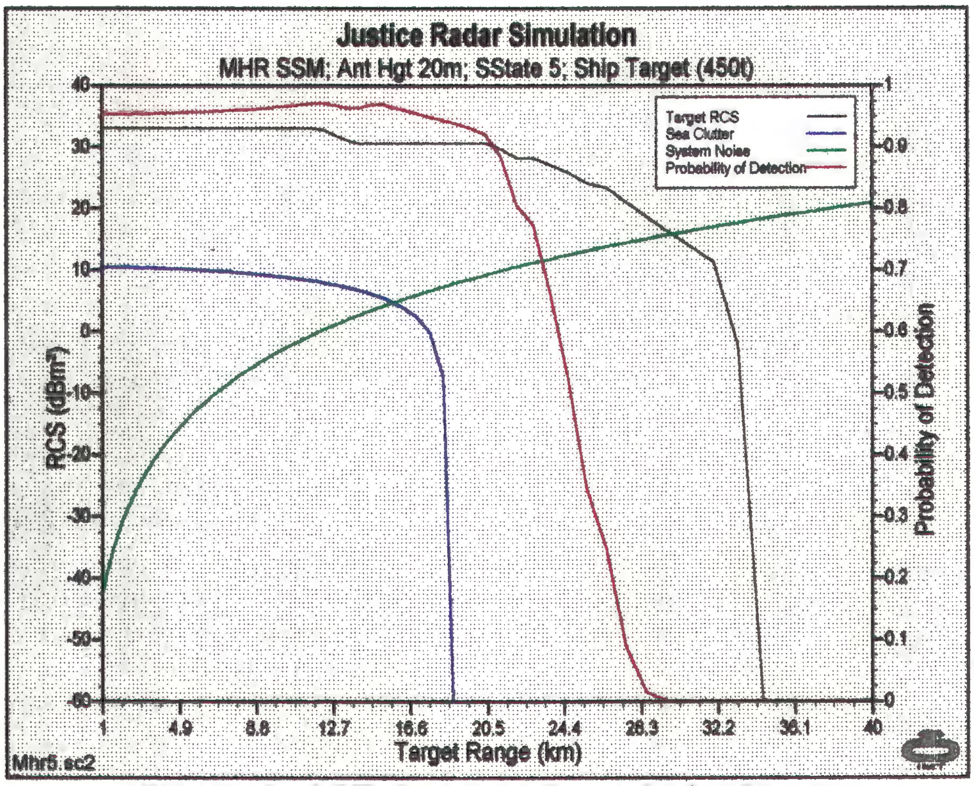


Figure D.14: MHR Simulation Output for Sea State 5.

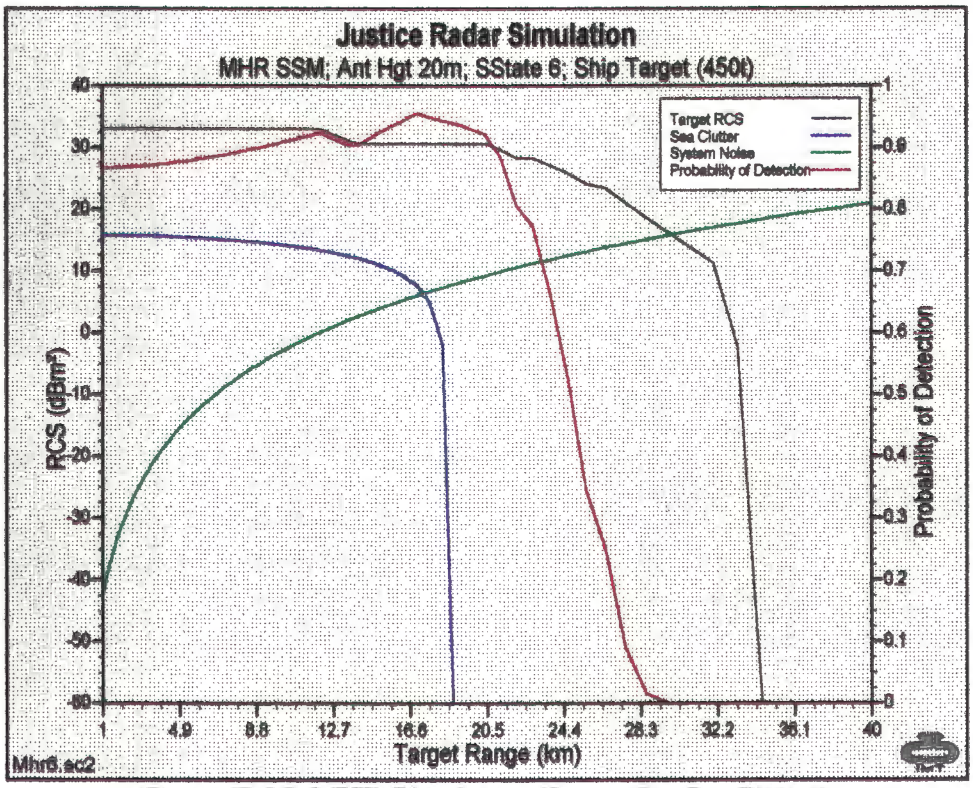


Figure D.15: MHR Simulation Output for Sea State 6.

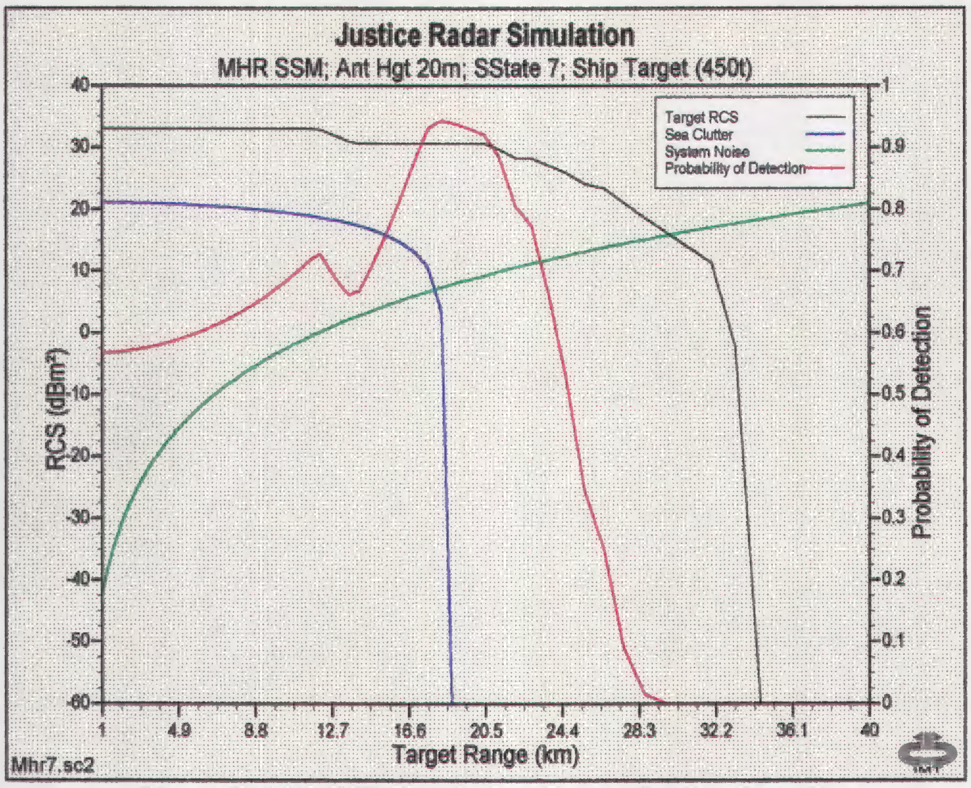


Figure D.16: MHR Simulation Output for Sea State 7.

Appendix E

DATA STRUCTURES AND SIMULATION VARIABLES

F.1 NON-STANDARD DATA STRUCTURES

F.1.1 SHIPTYPE

Record with the following fields:

- [posn] Ship's position. Store ship's current position.
- [head] Ship's heading. Store ship's current course in radians.
- [speed] Ship's speed. Store ship's current speed.
- [move] Ship's movement. Store ship's course and speed in vectortype.
- [rcsposn] Position of radar reflective areas relative to turning point in vectortype.
- [rcsmag] Magnitude of radar reflective area expressed as RCS for particular relative bearing.

F.1.2 VECTORTYPE

Array [2] of type real. The type allows Cartesian vectors to be stored where the first position is reserved for the east component and the second position is reserved for the north component of the Cartesian vector.

F.1.3 WINDTYPE

Record with the following fields:

- Array [36] of type real. This field contains the distribution function for wind direction in true compass degrees.
- Array [8,5] of type real. This field contains the eight distribution functions for wind speed in m/s for the probabilities 0, 0.25, 0.5, 0.75 and 1.

F.1.4 RADARTYPE

Array [2] of type real. This allows the complete radar detection distribution to be stored.

F.1.5 RCSPOSNTYPE

Array [17] of vectortype. Allows the storage of the seventeen radar reflective areas that make up the FAC(M).

F.1.6 RCSMAGTYPE

Array [36,17] of type real. Allows the storage of the radar cross section data for relative bearing and for the seventeen radar reflective areas that make up the FAC(M).

F.1.7 RANGEGATETYPE

Record with the following fields:

- [direction] Direction in which the MHR is pointing.
- [range] The range at which the gate is placed from the radar.
- [pos] The position of the range gate in the simulation grid.
- [right] Boolean value to guide angular movement of the range gate in the search phase.
- [up] Boolean value to guide the range movement of the range gate in the search phase.

F.1.8 SSMTYPE

Record with the following fields:

- [posn] SSM position in the simulation grid.
- [head] SSM heading in radians.
- [aim] Ordered heading for SSM.
- [gate] The SSM range gate which is of range gate type.
- [speed] SSM speed.
- [move] Vectortype that stores SSM movement vector in Cartesian grid co-ordinates.

F.1.9 STACKTYPE

Record with the following fields:

- [index] Index of position of last value stowed in the stack.
- [stack] array of type real to store information.

F.2 INTEGER VARIABLES CONTAINING RANDOM NUMBER STREAM REFERENCES

In order to facilitate the use of common random numbers for various runs of the simulation, the integer variables with self-explanatory names below, allow the use of a particular row of random numbers for a particular random process.

Variable Name	*Stream*
WindDirectionStream	1
WindSpeedStream	2
CurrentDirectionStream	3
CurrentVelocityStream	4
TemperatureStream	5
ShipsHeadStream	6
ShipsSpeedStream	7
SRDetectStream	8
SSMBrgStream	9
SSMRngStream	10
SSMHeadingSream	11
DetectRadarStream	12
MHRDetectStream	13
ESMbrgStream	14
CRCStream	15
MRCStream	16
TactABDelayStream	17
TactCDelayStream	18
TactDRangeStream	19
TactERangeStream	20

F.3 GLOBAL VARIABLES

Variable	*Type*	*Used for*
RCSship	array [1..17,1..3] of real	17 RCS positions with 2 grid data and one RCS measure
chafflist	array [1..80,1..3] of real	80 chaff RCS points with 2 grid data and one RCS value
Zrng	array [1..StrmNum] of longint	Seeds for random number generator

Variable	*Type*	*Used for*
CRate	array [11..15,1..9] of real	Course rate changes for speeds {22,24,26,28,30} every 5 seconds
CRCfTime	array[1..n] of real	Firing times for close range chaff
Acourse	boolean	Ship must alter course
ACQUIRE	boolean	Missile operating mode
Aspeed	boolean	Ship must increase speed to 30 knots
CRCPort	boolean	Fire CRC to Port
EOS	boolean	End-of-Simulation Replication
HOJ	boolean	Missile operating mode
JAMMER	boolean	Jammer state
LOCKED	boolean	Missile operating mode
MHR	boolean	MHR detected by ship
ReactionTimeElapsed	boolean	Reaction to ordered speed or course change has commenced
SEARCH	boolean	Missile operating mode
SRdetSSM	boolean	SSM detected by search radar
TacticOrdered	boolean	Order for FAC(M) to commence tactics to be employed
TurnPort	boolean	Alter course to port
AcqFactor	const	Range gate narrowing factor
AcqSS	const	Small range gate angular rate estimate
AcqSW	const	Large range gate angular rate estimate
AGate	const	Depth of tracking gate halves
BundleCRC	const	Mean RCS for one CRC bundle
BundleMRC	const	Mean RCS for one MRC bundle
CourseDiff	const	Decision constant
CRCats	const	Mean athwartships CRC bloom position
CRCdepR	const	SSM range at which CRC must be deployed
CRCdisp	const	CRC dispersion from bloom point
CRCint	const	Interval between firing subsequent CRC rockets
CRClong	const	Mean longitudinal CRC bloom position
currentA	const	Minimum Current
currentB	const	Most Likely Current
currentC	const	Maximum Current
ESMAdv	const	ESM advantage over radar
HalfBeam	const	Half SSM MHR beam width
hitdist	const	Hit criterion for SSM
InitTime	const	Initial time delay value

Variable	*Type*	*Used for*
Jamtime	const	Jamming time when so ordered and CRC not fired
LAGain	const	Lead angle gain
LSearchL	const	MHR lower range search limit
Mach	const	Missile speed
maxCRC	const	Number of CRC to be fired
maxrun	const	Number of Replications
MaxSAngle	const	Max left/right MHR search angle
maxspeed	const	FAC(M) maximum speed
MHRactRange	const	MHR activation range
MinRVal	const	Minimum SSM tracking range
MRCdepR	const	SSM range at which MRC must be deployed
MRCRange	const	Mean burst distance for MRC
MRCSD	const	SD for MRC deployment
MTR	const	SSM turning rate in m/s
prf	const	Pulse Repetition Frequency
R	const	Universal gas constant
RCSPts	const	Number of radar reflective points representing a FAC(M)
RelDir	const	Number of radar reflective bearings counting from zero
RHorizon	const	Radar Horizon
SearchGate	const	Depth of search gate halves
sidelobeR	const	Effective maximum side lobe range
SpeedDiff	const	Decision constant
Srate	const	Speed increase rate in m/s
SSMestSpeed	const	Estimated SSM speed
SSMhSD	const	SSM heading standard deviation
SSMmaxR	const	Maximum SSM run-in range
SSMminR	const	Minimum SSM run-in range
SStepA	const	MHR directional step size
SStepR	const	MHR range step size
Stacksize	const	Size of tracking stacks
StrmNum	const	Number of random streams
Tactic_A	const	Execute change Speed tactic
Tactic_B	const	Execute change Course tactic
Tactic_C	const	Execute jam MHR tactic
Tactic_D	const	Execute deploy CRC
Tactic_E	const	Execute deploy MRC
tempM	const	Mean temperature
tempSD	const	Temperature Standard Deviation
USearchL	const	MHR upper range search limit
y	const	Heat capacity/Pressure & Volume

Variable	*Type*	*Used for*
CRfile	file of real	Course rate changes for speeds {22,24,26,28,30} every 5 seconds
RadarDetectFile	file of real	Search Radar detection distribution
rcsmagfile	file of real	Magnitude data for radar reflective areas
rcsposfile	file of real	Positional data for radar reflective areas
windfile	file of windtype	WIND.DAT
ChaffCount	integer	Index for chafflist
ChaffNumber	integer	Number of CRC deployed
counter	integer	Counter
CRCindex	integer	counter
i	integer	Counter
j	integer	Counter
Mhit	integer	Record number of missile hits
RCSIndex	integer	SSM relative bearing from ship
RCSTest	integer	Test value for relative bearing change
SR	radartype	Radar Detection Distribution
AcqGate	real	Range gate depth for acquisition and tracking phases
Dtime	real	Simulation time step
GateWidth	real	Range gate time depth
Jtime	real	Time at which jamming must commence
meanAvector	real	Mean angular tracking rate
meanRvector	real	Mean range tracking rate
MHRDetect	real	MHR Elint detection range
MissDistance	real	SSM mean closest distance to FAC(M)s geometric centre
MRCfTime	real	Firing time for medium range chaff
NewCourse	real	Ordered new course
NewSpeed	real	Ordered new speed
RangeTest	real	Range FAC(M) to SSM
RSR	real	Radar detection range for SSM
Rtime	real	Reaction time to start course alteration and speed change
SSMBrg	real	SSM bearing from FAC(M)
SSMRange	real	SSM range from FAC(M)
Stime	real	Time elapsed since start of replication
TactABTime	real	Time at which Tactics A and B must commence

Variable	***Type***	***Used for***
TactCDETime	real	Time at which Tactics C, D and E must commence
temp	real	Temperature
TgtRelBrg	real	Target relative bearing
ship	shiptype	Ship data
SSM	SSMtype	SSM data
ErrorAStack	stack	Angular tracking errors
ErrorRStack	stack	Range tracking errors
reportfile	text file	Simulation Report
CurrentV	vectortype	Cartesian current vector
ShipCentre	vectortype	FAC(M) geometric centre position
WindV	vectortype	Cartesian wind vector
wind	windtype	Wind distributions

Appendix F

FUNCTIONS AND PROCEDURES

Procedure/Function	***Input***	***Output***
function CartForm	Degrees in compass co-ordinates	Degrees in Cartesian co-ordinates
function degree	radians	degrees
function degree	Radian	Degree
function DetectionRange	Empirical radar detection distribution	Radar detection range on SSM
function EmptyAStack		TRUE if stack is empty FALSE otherwise
function EmptyRStack		TRUE if stack is empty FALSE otherwise
function ESMbrg		SSM bearing by ESM
function FullAStack		TRUE if stack is full FALSE otherwise
function FullRStack		TRUE if stack is full FALSE otherwise
function logamp	x	$\log_{10}(x)$ if $x < 0$ 0 otherwise
function min	Two real numbers, a and b	$\min\{a,b\}$
function normal	Mean Standard Deviation Steam number	normal ~ $n(\bar{x}, s^2)$
function PopAError		Value of uppermost field in angle error stack
function PopRError		Value of uppermost field in range error stack
function power	x y	x^y
function radian	degrees	radians
function rand	Stream number	rand ~ U(0,1)

Procedure/Function	***Input***	***Output***
function Randgt	Stream number	Seed associated with stream number
function tan	angle in radians	tangent of the angle
function triang	Lower limit Upper limit Most likely value Stream number	triang ~ triang(a,b,c)
function turnrate		Ship's turning rate at a given time
function uniform	Lower limit Upper limit Stream number	uniform ~ U(LL,UL)
function weibull	Parameter α_1 Parameter α_2 Stream number	weibull ~ weibull(α_1, α_2)
procedure SendAcquirePulse		Simulated acquisition pulse
procedure CalculateAngleError	Half width of tracking beam Integrated values for both beam halves	Angle Tracking Error in radians
procedure CalculateRangeError	Early Gate size Late Gate size Integrated values for both gates	Range Tracking Error in m/s
procedure CalculateVector	Distance part of polar vector Angle part of polar vector	Cartesian vector
procedure ExecuteTactics		Ship's chosen tactics enabled
procedure FinalReport		Print final report to screen and report file
procedure FireCRC		Simulated firing of one CRC rocket
procedure FireMRC		Simulated firing of eight MRC rockets
procedure GenerateCurrent		Sea current in Cartesian co-ordinates
procedure GenerateWindVector		Wind vector in Cartesian co-ordinates
procedure GetWindDirection		Wind direction in degrees
procedure GetWindSpeed	Wind direction in degrees	Wind speed in knots
procedure InitAErrorStack		Angle error stack initialised

Procedure/Function	***Input***	***Output***
procedure Initialise-GlobalVariables Replication		All global variables for a particular replication initialised
procedure Initialise-GlobalVariables Simulation		All global variables for simulation initialised
procedure InitRErrorStack		Range error stack initialised
procedure PushAError	x	x in top of angle error stack
procedure PushRError	x	x in top of range error stack
procedure Randdf		Initialise seed values in list Zrng
procedure Randst	New seed number Stream number	Update seed values in list Zrng
procedure ResolveRangeGate	Early gate time Late gate time	Integrated range values Integrated angle values
procedure SendSearchPulse		Simulated search pulse
procedure SendTrackPulse		Simulated tracking pulse
procedure SetSearchGate	Maximum MHR offset Initial search distance	Search gate set to commence search
procedure ShipDirectionFind	SSM position in the grid	Ship's bearing from the SSM in radians
procedure ShipModel		Simulated ship actions for one pulse interval
procedure ShipRCSMag		Ship's RCS values for radar reflective points and SSM relative bearing
procedure ShipRCSPlan		Ship's radar reflective points and ship's centre position in Cartesian co-ordinates
procedure SSMAlterCourse		SSM course after next simulation increment
procedure SSMModel		Simulated SSM actions for one pulse interval
procedure SSMRelBrg	SSM bearing Ship's course	Relative bearing index for RCS magnitude look-up table
procedure transform	Radar reflective item posn Range gate posn	Radar reflective item posn in range gate co-ordinates
procedure VerifyEOS		Termination of simulation replication verified

Appendix G

COMPUTER SOURCE CODE

The program source code is overleaf.

```
program simulation;
(*  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *)
(* Program to simulate various counter-missile tactics in order to *)
(* determine the effectiveness of such tactics in terms of missile *)
(* hit probability and miss distance.                              *)
(*  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *)
(* Version 1.10                                                    *)
(*  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *)
(* Programmed in Turbo Pascal 7.0                                  *)
(*  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *)
(* Capt G.N. Engelbrecht, S.A. Navy                                *)
(*  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *)
(* The following files must be in the default directory :         *)
(*     WIND    .DAT   : Wind distribution data                     *)
(*     RCSMAG  .DAT   : RCS magnitude data for ship                *)
(*     RCSPOS  .DAT   : RCS relative positional data for ship      *)
(*     ALTCO   .DAT   : Course rate change for altering course     *)
(*     RADAR   .DAT   : Radar Prediction Distribution              *)
(*  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *)

{---}  {--------------------}
{$N+}  { Compiler Directive }
{---}  {--------------------}

uses crt;
                                     {----------------------------------}
const maxrun      =        3;        { Number of replications (115)     }
      StrmNum     =       20;        { Number of random streams         }
      maxspeed    =     30.0;        { Ship's maximum speed             }
      prf         =      500;        { Missile Radar prf                }
      tempM       =     21.7;        { Mean temperature                 }
      tempSD      =    2.756;        { Temperature standard deviation   }
      currentA    =    0.384;        { Minimum current                  }
      currentB    =    0.768;        { Most likely current              }
      currentC    =    2.560;        { Maximum Current                  }
      SSMminR     =       12;        { Minimum SSM run in range         }
      SSMmaxR     =       16;        { Maximum SSM run in range         }
      SSMhSD      =     1.67;        { SSM heading SD in degrees        }
      SSMestSp    =    310.0;        { Estimated SSM speed              }
      R           =   287.06;        { Universal gas constant           }
      y           =      1.4;        { heat capacity/pressure & volume  }
      Mach        =      0.9;        { Mach Number for missile          }
      SpeedDiff   =     1E-5;        { Speed difference constant        }
      CourseDiff  =     1E-3;        { Speed difference constant        }
      SRate       =   0.2048;        { Speed change rate in m/s         }
      RCSPts      =       17;        { Number of radar points on FAC(M) }
      RelDir      =       35;        { Number of radar relative bearings}
      RHorizon    =    35125;        { Radar Horizon                    }
      ESMAdv      =      1.4;        { ESM advantage over Radar         }
      JamTime     =      3.0;        { Jam time when not firing CRC     }
      CRClong     =     40.0;        { Mean Longitudinal CRC bloom posn }
      CRCats      =      9.0;        { Mean Athwartships CRC bloom posn }
      CRCdisp     =      2.0;        { CRC dispersion from bloom point  }
      maxCRC      =        2;        { Number of CRC to be fired        }
      BundleCRC   =    100.0;        { Mean RCS for one CRC bundle      }
      CRCdepR     =  12000.0;        { SSM range to deploy CRC          }
      CRCint      =      1.0;        { Interval between CRC firings     }
      MRCdepR     =  14000.0;        { SSM range to deploy MRC          }
      MRCRnge     =   1000.0;        { Mean distance to burst for MRC   }
      BundleMRC   =     80.0;        { Mean RCS for one MRC bundle      }
      MRCSD       =      1.0;        { MRC SD for deployment            }
      MRCnum      =        8;        { Number of MRC rockets deployed   }
      MRCbundles  =        3;        { Number of bundles in MRC rocket  }
      MHRactRange=   22000.0;        { MHR activation range             }
      MaxSAngle   =     25.0;        { Max left/right MHR search angle  }
```

```
        SStepA     =      1.0;      { MHR directional search step size }
        SStepR     =      2.0;      { MHR range search step size       }
        USearchL   =  12000.0;      { MHR upper range search limit     }
        LSearchL   =   6000.0;      { MHR lower range search limit     }
        SearchGate =    500.0;      { Depth of the search gate halves  }
        AGate      =     40.0;      { Depth of tracking gate halves    }
        MinRVal    =    150.0;      { Minimum range tracking distance  }
        Halfbeam   =     0.55;      { Half MHR beamwidth               }
        sidelobeR  =   2000.0;      { Effective side lobe range        }
        AcqFactor  =     0.55;      { RG narrowing acquisition factor  }
        AcqSW      =   0.0010;      { Large RG angular rate estimate   }
        AcqSS      =   0.0005;      { Small RG angular rate estimate   }
        Stacksize  =      400;      { Number of pulses integrated      }
        LAGain     =     0.12;      { Lead Angle Gain                  }
        MTR        =     70.0;      { SSM turning rate in degrees/s    }
        hitdist    =     31.0;      { Hit criterion for SSM            }
        InitTime   =    500.0;      { Initial time delay value         }
        Tactic_A   =    FALSE;      { Change speed tactic              }
        Tactic_B   =    FALSE;      { Change course tactic             }
        Tactic_C   =    FALSE;      { Jam MHR tactic                   }
        Tactic_D   =    FALSE;      { Deploy CRC tactic                }
        Tactic_E   =    FALSE;      { Deploy MRC tactic                }
                                    {----------------------------------}

type    windtype    = record
                         direction  : array [1..36] of real;
                         speed      : array [1..8,1..5] of real;
                      end;

        vectortype  = array[1..2] of real;

        rcsposntype = array [1..RCSPts] of vectortype;

        rcsmagtype  = array [0..RelDir,1..RCSPts] of real;

        radartype   = array [1..5] of real;

        shiptype    = record
                         posn    : vectortype;
                         head    : real;
                         speed   : real;
                         move    : vectortype;
                         rcsposn : rcsposntype;
                         rcsmag  : rcsmagtype;
                      end;

        RangeGateType = record
                           direction,
                           range     : real;
                           pos       : vectortype;
                           right     : boolean;
                           up        : boolean;
                        end;

        SSMtype     = record
                         posn      : vectortype;
                         head      : real;
                         aim       : real;
                         gate      : RangeGateType;
                         speed     : real;
                         move      : vectortype;
                      end;
```

```
      stacktype  = record
                      index : integer;
                      stack : array[1..Stacksize] of real;
                   end;

var   Zrng                     : array[1..StrmNum] of longint;
      CRate                    : array[11..15,1..9] of real;
      RCSship                  : array[1..RCSPts,1..3] of real;
      chafflist                : array[1..80,1..3] of real;
      CRCfTime                 : array[1..maxCRC] of real;
      windfile                 : file of windtype;
      rcsposfile,
      rcsmagfile,
      CRfile                   : file of real;
      RadarDetectFile          : file of real;
      reportfile               : text;
      wind                     : windtype;
      ship                     : shiptype;
      SSM                      : SSMtype;
      SR                       : radartype;
      ShipCentre,
      WindV,
      CurrentV                 : vectortype;
      ErrorRStack,
      ErrorAStack              : stacktype;
      WindDirectionStream,
      WindSpeedStream,
      CurrentDirectionStream,
      CurrentVelocityStream,
      TemperatureStream,
      ShipsHeadStream,
      ShipsSpeedStream,
      SRDetectStream,
      SSMBrgStream,
      SSMRngStream,
      SSMHeadingStream,
      DetectRadarStream,
      MHRDetectStream,
      ESMbrgStream,
      CRCStream,
      MRCStream,
      TactABDelayStream,
      TactCDelayStream,
      TactDRangeStream,
      TactERangeStream,
      Mhit,
      counter,
      RCSIndex,
      RCSTest,
      CRCindex,
      ChaffNumber,
      ChaffCount,
      i,
      j                        : integer;
      temp,
      Dtime,
      Stime,
      Rtime,
      MissDistance,
      meanAvector,
      meanRvector,
      GateWidth,
      RSR,
      SSMBrg,
      SSMRange,
```

```
      NewSpeed,
      NewCourse,
      MHRDetect,
      RangeTest,
      AcqGate,
      TactABTime,
      TactCDETime,
      MRCfTime,
      JTime,
      TgtRelBrg              : real;
      EOS,
      SEARCH,
      ACQUIRE,
      LOCKED,
      HOJ,
      MHR,
      JAM,
      JAMMER,
      ACourse,
      Aspeed,
      SRdetSSM,
      ReactionTimeElapsed,
      TurnPort,
      CRCPort,
      TacticOrdered,
      PURSUIT,
      LEADANGLE,
      BALLISTIC,
      UpdateAim,
      CRCdeploy,
      MRCdeploy              : boolean;

procedure Randdf;
{-------------------------------------------------}
{ Set the seeds for all 100 random number streams }
{-------------------------------------------------}
begin {Randdf}
   Zrng[ 1] := 1973272912;
   Zrng[ 2] :=  281629770;
   Zrng[ 3] :=   20006270;
   Zrng[ 4] := 1280689831;
   Zrng[ 5] := 2096730329;
   Zrng[ 6] := 1933576050;
   Zrng[ 7] :=  913566091;
   Zrng[ 8] :=  246780520;
   Zrng[ 9] := 1363774876;
   Zrng[10] :=  604901985;
   Zrng[11] := 1511192140;
   Zrng[12] := 1259851944;
   Zrng[13] :=  824064364;
   Zrng[14] :=  150493284;
   Zrng[15] :=  242708531;
   Zrng[16] :=   75253171;
   Zrng[17] := 1202299975;
   Zrng[18] := 1964472944;
   Zrng[19] :=  233217322;
   Zrng[20] := 1911216000;
end;   {Randdf}
```

```
function rand (Stream : integer) : real;
   {-----------------------------------------------}
   { Generate a random number for a specific stream }
   {-----------------------------------------------}
   { Define the constants }
   {---------------------}
   const    B2E15  =      32768;
            B2E16  =      65536;
            Modlus = 2147483647;
            Mult1  =      24112;
            Mult2  =      26143;

   var      Hi15,Hi31,Low15,Lowprd,Ovflow,zi : longint;

begin   {Rand}
   {--------------------------------------------------------}
   { Generate the next random number for a specific stream }
   {--------------------------------------------------------}
   Zi     := Zrng[Stream];
   Hi15   := Zi DIV B2E16;
   Lowprd := (Zi - Hi15 * B2E16) * Mult1;
   Low15  := Lowprd DIV B2E16;
   Hi31   := Hi15 * Mult1 + Low15;
   Ovflow := Hi31 DIV B2E15;
   Zi     := (((Lowprd - Low15 * B2E16) - Modlus) +
               (Hi31 - Ovflow * B2E15) * B2E16) + Ovflow;
   if Zi < 0 then Zi := Zi + Modlus;
   Hi15   := Zi DIV B2E16;
   Lowprd := (Zi -Hi15 * B2E16) * Mult2;
   Low15  := Lowprd DIV B2E16;
   Hi31   := Hi15 * Mult2 + Low15;
   Ovflow := Hi31 DIV B2E15;
   Zi     := (((Lowprd - Low15 * B2E16) - Modlus) +
               (Hi31 - Ovflow * B2E15)* B2E16) + Ovflow;
   if Zi < 0 then Zi := Zi + Modlus;
   Zrng[Stream] := Zi;
   Rand   := (2 * (Zi DIV 256) + 1) / 16777216.0;
end;    {Rand}

procedure Randst (Zset : integer; Stream : Integer);
{-----------------------------------------------------------}
{ Procedure to update seeds for random number generator }
{-----------------------------------------------------------}
begin    {Randst}
   Zrng[Stream] := Zset;
end;     {Randst}

function Randgt (Stream : integer) : longint;
{-------------------------------------------}
{ Function to obtain seed for random number }
{ generator from seed list                  }
{-------------------------------------------}
begin   {Randgt}
   Randgt := Zrng[Stream];
end;    {Randgt}

function tan (angle : real) : real;
{----------------------------}
{ Return the tan of an angle }
{----------------------------}
begin
   tan := sin(angle)/cos(angle);
end;
```

```
function power(x,y : real) : real;
{--------------}
{ Returns x^y  }
{--------------}
begin
   if y > 0.0 then
      begin
         power := exp(ln(x) * y);
      end
   else
      begin
         if y = 0 then
            begin
               power := 1
            end
         else
            begin
               power := 1/(exp(ln(x) * abs(y)));
            end;
      end;
   end;

function logamp(x : real) : real;
{------------------------------------}
{ Simulates a logarithmic amplifier }
{------------------------------------}
begin
   if x < 1 then
      begin
         logamp := 0;
      end
   else
      begin
         logamp := ln(x) + 0.434294482;
      end;
end;

function min(a,b : real) : real;
{------------------------------------------}
{ Returns the minimum of two real numbers }
{------------------------------------------}
begin
   if a<b then
      begin
         min := a;
      end
   else
      begin
         min := b;
      end;
end;

function radian(deg : real) : real;
{--------------------------------------------}
{ Function to transform degrees into radians }
{--------------------------------------------}
begin
   radian := deg*(2*pi)/360;
end;
```

```
function degree (rad : real) : real;
{-------------------------------------------}
{ Function to transform radians into degrees }
{-------------------------------------------}
begin
   degree := rad*360/(2*pi);
end;

function CartForm(DegreeTrue: real) : real;
{-----------------------------------------------------}
{ Function to transform angle from Polar to Cartesian }
{-----------------------------------------------------}
var q : real;

begin
   q := 360 - DegreeTrue + 90;
   if q > 360 then
      begin
        q := q - 360;
      end;
   CartForm := q;
end;

procedure SSMRelBrg(brg, hdg : real ; var index : integer);
{------------------------------------}
{ Return index to look up ship's RCS }
{------------------------------------}
var value : real;

begin
   value := brg - hdg;
   if value<0 then
      begin
         value := value + 360;
      end;
   index := trunc (value/10 + 0.5);
   if index = 36 then
      begin
         index := 0;
      end;
end;

function EmptyRStack  : boolean;
{--------------------------------}
{ Returns TRUE if stack is empty }
{--------------------------------}
begin
   if ErrorRStack.index = 0 then
      begin
         EmptyRStack := TRUE;
      end
   else
      begin
         EmptyRStack := False;
      end;
end;
```

```
function FullRStack : boolean;
{--------------------------------}
{ Returns TRUE if stack is full }
{--------------------------------}
begin
   if ErrorRStack.index = Stacksize then
      begin
         FullRStack := TRUE;
      end
   else
      begin
         FullRStack := FALSE;
      end;
end;

procedure InitRErrorStack;
{-------------------------------------}
{ Procedure to initialise ErrorStack }
{-------------------------------------}
begin
   ErrorRStack.index := 0;
end;

procedure PushRError (val : real);
{---------------------------------------------}
{ Procedure to stow gate error in Errorstack }
{---------------------------------------------}
begin
   with ErrorRStack do
      begin
         index        := index + 1;
         stack[index] := val;
      end;
end;

function PopRError   : real;
{--------------------------------------------------}
{ Procedure to extract gate error from Errorstack }
{--------------------------------------------------}
begin
   with ErrorRStack do
      begin
         PopRError   := stack[index];
         index       := index - 1;
      end;
end;

function EmptyAStack  : boolean;
{---------------------------------}
{ Returns TRUE if stack is empty }
{---------------------------------}
begin
   if ErrorAStack.index = 0 then
      begin
         EmptyAStack := TRUE;
      end
   else
      begin
         EmptyAStack := False;
      end;
end;
```

```
function FullAStack : boolean;
{-------------------------------}
{ Returns TRUE if stack is full }
{-------------------------------}
begin
   if ErrorAStack.index = Stacksize then
      begin
         FullAStack := TRUE;
      end
   else
      begin
         FullAStack := FALSE;
      end;
end;

procedure InitAErrorStack;
{------------------------------------}
{ Procedure to initialise ErrorStack }
{------------------------------------}
begin
   ErrorAStack.index := 0;
end;

procedure PushAError (val : real);
{---------------------------------------------}
{ Procedure to stow gate error in Errorstack }
{---------------------------------------------}
begin
   with ErrorAStack do
      begin
         index        := index + 1;
         stack[index] := val;
      end;
end;

function PopAError : real;
{---------------------------------------------------}
{ Procedure to extract gate error from Errorstack }
{---------------------------------------------------}
begin
   with ErrorAStack do
      begin
         PopAError   := stack[index];
         index       := index - 1;
      end;
end;

function normal (x,s : real; stream : integer) : real;
{----------------------------------------------------------}
{ Returns a normal distributed variate by the polar method }
{----------------------------------------------------------}
var U1, U2, V1, V2, W, Y : real;
begin
   W  := 2;
   while W>1 do
      begin
         U1 := rand(stream);
         U2 := rand(stream);
         V1 := 2*U1-1;
         V2 := 2*U2-1;
         W  := sqr(V1) + sqr(V2);
      end;
   Y      := V1 * sqrt(-2*ln(W)/W);
   normal := x + s*Y;
end;
```

```
function uniform(a,b : real; stream : integer) : real;
{---------------------------------------------}
{ Returns a uniform distributed random number }
{---------------------------------------------}
   begin
      uniform := a + (b-a)*rand(stream);
   end;

function triang(a,b,c : real; stream : integer) : real;
{------------------------------------------------}
{ Returns a Triangular-distributed random number }
{------------------------------------------------}
var U,X,c1 : real;

begin
   U  := rand(stream);
   c1 := (c-a)/(b-a);
   if NOT(U>c1) then
      begin
         X      := sqrt(U*c1);
      end
   else
      begin
         X :=1 - sqrt((1-c1)*(1-U));
      end;
   triang := a + (b-a)*X;
end;

function weibull(a : real; stream : integer) : real;
{---------------------------------------------}
{ Returns a weibull distributed random number }
{---------------------------------------------}
begin
   weibull := power(-ln(rand(stream)),1/a);
end;

function ESMbrg : real;
{-------------------------------------------------------------}
{ Returns the ESM bearing of an active radar source detected }
{-------------------------------------------------------------}
var b,
    x,
    y : real;

begin
   x := SSM.posn[1] - ship.posn[1];
   y := SSM.posn[2] - ship.posn[2];
   if x <> 0 then
      begin
         if x<0 then
            begin
               b := pi + arctan(y/x);
            end
         else
            begin
               if y<0 then
                  begin
                     b := 2*pi + arctan(y/x);
                  end
```

```
               else
                  begin
                     b := arctan(y/x);
                  end;
            end;
      end;
   b := degree(b);
   ESMbrg := trunc(uniform(b-2,b+2,ESMbrgStream));
end;

function DetectionRange(R : radartype):real;
{------------------------------}
{ Return Radar Detection Range }
{------------------------------}
var U,
    P : real;

begin
   U               := rand(DetectRadarStream);
   P               := 4 * U;
   J               := trunc(P) + 1;
   DetectionRange := R[J] + (P - J + 1)*(R[J+1] - R[J]);
end;

function turnrate : real;
{--------------------------------------------}
{ Return the turning rate of the ship in m/s }
{--------------------------------------------}
var T,
    S        : integer;

begin
   {---------------------------------------------------}
   { Find index for time elapsed since beginning of turn }
   {---------------------------------------------------}
   T := trunc((Stime - Rtime)/5)+1;
   {-----------------}
   { Restrict T to 9 }
   {-----------------}
   if T>9 then
      begin
         T := 9;
      end;
   {------------------------------}
   { Find index for present speed }
   {------------------------------}
   S := trunc((ship.speed/0.512)/2 + 0.5);
   {----------------------}
   { Look up turning rate }
   {----------------------}
   turnrate := radian(CRate[S,T]);
end;

procedure FireCRC;
{----------------------------------------------------------}
{ Procedure to simulate the firing of a single CRC rocket }
{----------------------------------------------------------}
var point     : vectortype;
    anchor,
    angle     : real;
    relangle,
    index     : integer;
```

```
begin
   {-----------------------------}
   { Increment CRC Chaff counter }
   {-----------------------------}
   ChaffNumber := ChaffNumber + 1;
   {-----------------}
   { Find firing side }
   {-----------------}
   SSMRelBrg(SSMBrg,degree(ship.head),RCSIndex);
   relangle := RCSIndex + 1;
   if (relangle > 0) AND (relangle < 18) then
      begin
         CRCPort := TRUE;
      end
   else
      begin
         CRCPort := FALSE;
      end;
   {----------------------------------}
   { Set mean relative bloom position }
   {----------------------------------}
   if CRCPort then
      begin
         angle := ship.head + arctan(CRCats/CRClong);
      end
   else
      begin
         angle := ship.head - arctan(CRCats/CRClong);
      end;
   if angle < 0 then
      begin
         angle := angle + 2*pi;
      end;
   if angle > 2*pi then
      begin
         angle := angle - 2*pi;
      end;
   point[1] := ship.posn[1] + sqrt(sqr(CRCats)+sqr(CRClong))
               * cos(angle);
   point[2] := ship.posn[2] + sqrt(sqr(CRCats)+sqr(CRClong))
               * sin(angle);
   {-----------------------------------------------------------}
   { Place bloom positions for three bundles in Cartesian grid }
   {-----------------------------------------------------------}
   for index := 1 to 3 do
      begin
         ChaffCount := ChaffCount + 1;
         anchor := radian(uniform(0,360,CRCStream));
         chafflist[ChaffCount,1] := point[1] + CRCdisp*cos(anchor);
         chafflist[ChaffCount,2] := point[2] + CRCdisp*sin(anchor);
         chafflist[ChaffCount,3] := BundleCRC;
      end
end;
```

```
procedure FireMRC;
{-------------------------------------}
{ Procedure to fire a pattern of MRC }
{-------------------------------------}
var posn    : vectortype;
    bearing : real;
    index,
    n       : integer;

begin
   bearing     := ship.head;
   for index := 1 to MRCnum do
      begin
         bearing := bearing + radian(45);
         if bearing>2*pi then
            begin
               bearing := bearing - 2*pi;
            end;
         for n := 1 to MRCbundles do
            begin
               ChaffCount := ChaffCount + 1;
               posn[1] := ship.posn[1] + MRCRnge*cos(bearing);
               posn[2] := ship.posn[2] + MRCRnge*sin(bearing);
               chafflist[ChaffCount,1] := normal(posn[1], MRCSD,
                                                 MRCStream);
               chafflist[ChaffCount,2] := normal(posn[2],MRCSD,
                                                 MRCStream);
               chafflist[ChaffCount,3] := BundleMRC;
            end;
      end;
end;


procedure CalculateRangeError (EG,LG  : real;
                               TG     : vectortype;
                               var RE : real);
{-----------------------------------------------------------}
{ Procedure to calculate the range error of the range gate }
{-----------------------------------------------------------}
var delta : real;

begin
   {----------------------------}
   { Logarithmic Amplification }
   {----------------------------}
   if NOT(SEARCH) then
      begin
         TG[1] := 10 * logamp(TG[1]);
         TG[2] := 10 * logamp(TG[2]);
      end;
   {-----------------}
   { Calculate Error }
   {-----------------}
   delta  := (TG[2]-TG[1])/2;
   if TG[1]>TG[2] then
      begin
         RE := (delta/TG[1])*EG;
      end
   else
      begin
         RE := (delta/TG[2])*LG;
      end;
end;
```

```
procedure CalculateAngleError (AG     : vectortype;
                               var AE : real);
{-----------------------------------------------------------}
{ Procedure to calculate the range error of the range gate }
{-----------------------------------------------------------}
var delta   : real;

begin
   {-----------------}
   { Calculate Error }
   {-----------------}
   delta  := AG[1]-AG[2];
   if (AG[1]>0) AND (AG[2]>0) then
      begin
        if AG[1]>AG[2] then
           begin
              AE := AcqSS;
           end
        else
           begin
              AE := -AcqSS;
           end;
      end
   else
      begin
        if AG[1]>0 then
           begin
              AE := AcqSW;
           end
        else
           begin
              AE := -AcqSW;
           end;
      end;
end;

procedure transform (rpos1,rpos2 : real;
                     gpos : vectortype;
                     gdir : real;
                     var tpos : vectortype);
{----------------------------------------------------------------}
{ Procedure to transform points in the standard Cartesian grid }
{ to points in the range gate grid                               }
{----------------------------------------------------------------}
var intdir,
    range,
    cdir : real;

begin
   {-----------------}
   { Transform entity }
   {--------------------------------------------}
   { Find range from range gate centre to entity }
   {--------------------------------------------}
   range  := sqrt(sqr(rpos1-gpos[1]) + sqr(rpos2-gpos[2]));
```

```
   {-----------------------------------------------------}
   { Calculate entity direction from range gate centre }
   {-----------------------------------------------------}
   if rpos1=gpos[1] then
      begin
         if rpos2<gpos[2] then
            begin
               intdir := radian(90);
            end
         else
            begin
               intdir := radian(270);
            end;
      end
   else
      begin
         if rpos1>gpos[1] then
            begin
               intdir := pi + arctan((rpos2-gpos[2])/
                         (rpos1-gpos[1]));
            end
        else
            begin
               intdir := arctan((rpos2-gpos[2])/(rpos1-gpos[1]));
               if intdir<0 then
                  begin
                     intdir := intdir + 2*pi;
                  end;
            end;
      end;
   intdir := intdir - pi;
   if intdir < 0 then
      begin
         intdir := intdir + 2*pi;
      end;
   cdir := intdir - gdir;
   if cdir < 0 then
      begin
         cdir := cdir + 2*pi;
      end;
   {-----------------------------------------------}
   { Calculate target position in range gate grid }
   {-----------------------------------------------}
   tpos[1] := range*cos(cdir);
   tpos[2] := range*sin(cdir);
end;

Procedure ResolveRangeGate (Etime,Ltime : real;
                            var TimeG, AngleG : vectortype);
{----------------------------------------------------------------}
{ Procedure to find radar reflective surfaces in the range gate }
{ and integrate the early and late gates                        }
{----------------------------------------------------------------}

type  itemtype = array[1..3] of real;

      gatestacktype    = record
                            index : integer;
                            data  : array[1..65] of itemtype;
                         end;

var factor,
    bearing,
    dist       : real;
    intpos,
```

```
    cpos       : vectortype;
    count      : integer;
    reflect    : itemtype;
    stack      : gatestacktype;

    function EmptyStack (test : integer) : boolean;
    {-------------------------------}
    { Function to check stack empty }
    {-------------------------------}
       begin
          if test = 0 then
             begin
                EmptyStack := TRUE;
             end
          else
             begin
                EmptyStack := FALSE;
             end;
       end;

    procedure pop (var item : itemtype);
    {---------------------------------------}
    { Procedure to extract item from stack }
    {---------------------------------------}
       begin
          with stack do
             begin
                item[1] := data[index,1];
                item[2] := data[index,2];
                item[3] := data[index,3];
                index := index - 1;
             end;
       end;

    procedure push ( item : itemtype);
    {----------------------------------}
    { Procedure to stow item in stack }
    {----------------------------------}
       begin
          with stack do
             begin
                index := index + 1;
                data[index,1] := item[1];
                data[index,2] := item[2];
                data[index,3] := item[3];
             end;
       end;

begin
   {------------------}
   { Initialise Gates }
   {--------------------------------}
   TimeG[1]  := 0;    { Early Gate }
   TimeG[2]  := 0;    { Late Gate  }
   AngleG[1] := 0;    { Right Gate }
   AngleG[2] := 0;    { Left Gate  }
   {--------------------------------}
   { Initialise stack }
   {------------------}
   stack.index := 0;
   {----------------------------------}
   { Set leading edge tracking factor }
   {----------------------------------}
   factor := Ltime/Etime;
```

```
{-------------------------------------------}
{ Transform all radar reflective points into }
{ gate area co-ordinates and push in stack   }
{-------------------------------------------}
with SSM.gate do
   begin
      pos[1] := SSM.posn[1] + range*cos(direction);
      pos[2] := SSM.posn[2] + range*sin(direction);
      {----------------}
      { Transform ship }
      {----------------}
      for count := 1 to RCSPts do
         begin
            transform(RCSship[count,1],RCSship[count,2],
                      pos,direction,intpos);
            reflect[1] := intpos[1];
            reflect[2] := intpos[2];
            reflect[3] := ship.rcsmag[RCSIndex,count];
            push(reflect);
         end;
      {-----------------}
      { Transform Chaff }
      {-----------------}
      if ChaffCount>0 then
         begin
            for count:= 1 to ChaffCount do
               begin
                  cpos[1] := chafflist[count,1];
                  cpos[2] := chafflist[count,2];
                  transform(cpos[1],cpos[2],pos,direction,intpos);
                  reflect[1] := intpos[1];
                  reflect[2] := intpos[2];
                  reflect[3] := chafflist[count,3];
                  push(reflect);
               end;
         end;
      {----------------------}
      { Integrate Range Gate }
      {----------------------}
      while NOT(EmptyStack(stack.index)) do
         begin
            {------------------------------}
            { Recover radar reflective item }
            {------------------------------}
            pop(reflect);
            {-------------------------------------}
            { Check radar reflective item in gate }
            {-------------------------------------}
            if (reflect[1]<(Ltime)) AND
               (reflect[1]>(-Etime)) AND
               (reflect[2]<(GateWidth)) AND
               (reflect[2]>(-GateWidth)) then
               begin
                  if reflect[1]>0 then
                     begin
                        {-------------------}
                        { Item in late gate }
                        {-------------------}
                        TimeG[2] := TimeG[2] + reflect[3];
                     end
```

```
                    else
                       begin
                          {--------------------}
                          { Item in early gate }
                          {--------------------}
                          TimeG[1] := TimeG[1] + factor*reflect[3];
                       end;
                    if reflect[2]>0 then
                       begin
                          {--------------------}
                          { Item in right gate }
                          {--------------------}
                          AngleG[1] := AngleG[1] + reflect[3];
                       end
                    else
                       begin
                          {-------------------}
                          { Item in left gate }
                          {-------------------}
                          AngleG[2] := AngleG[2] + reflect[3];
                       end;
                 end;
              end;
        end;
end;

Procedure SetSearchGate (angle,value : real);
{----------------------------------------------------------------}
{ Procedure to set up the initial position of the MHR range gate }
{----------------------------------------------------------------}
begin
   with SSM.gate do
      begin
         direction := SSM.head + radian(angle/2);
         if direction > 2*pi then
            begin
               direction := direction - 2*pi;
            end;
         range     := value;
         GateWidth := range * tan(radian(Halfbeam));
         right     := TRUE;
         up        := TRUE;
      end;
end;

procedure SendSearchPulse;
{----------------------------------------}
{ Procedure to simulate a MHR search pulse }
{----------------------------------------}
var testvalue,
    AError,
    RError        : real;
    TimeG,
    AngleG        : vectortype;
```

```
begin
   with SSM.gate do
      begin
         {---------------------------}
         { Set up angular test value }
         {---------------------------}
         if ((SSM.head>radian(360-MaxSAngle))AND(direction<pi)) then
            begin
               testvalue := 2*pi - SSM.head + direction;
            end
         else
            begin
               if ((SSM.head<radian(MaxSAngle))AND(direction>pi))then
                  begin
                     testvalue := 2*pi - direction + SSM.head;
                  end
               else
                  begin
                     testvalue := abs(SSM.head - direction);
                  end;
            end;
         {---------------------------------------}
         { If at limit, change search direction }
         {---------------------------------------}
         if testvalue>radian(MaxSAngle) then
            begin
               if right=TRUE then
                  begin
                     right := FALSE;
                  end
               else
                  begin
                     right := TRUE;
                  end;
            end;
         {----------------------------}
         { Increment search direction }
         {----------------------------}
         if right then
            begin
               direction := direction - radian(SStepA);
               if direction < 0 then
                  begin
                     direction := direction + 2*pi;
                  end;
            end
         else
            begin
               direction := direction + radian(SStepA);
               if direction > 2*pi then
                  begin
                     direction := direction - 2*pi;
                  end;
            end;
         {-----------------------------}
         { Set range gate centre range }
         {-----------------------------}
         if range < UsearchL then
            begin
               range := range + SStepR;
            end
         else
            begin
               range := LsearchL;
            end;
```

```
        {---------------------}
        { Set half beam width }
        {---------------------}
        GateWidth := range * tan(radian(Halfbeam));
        {--------------------}
        { Resolve Range Gate }
        {--------------------}
        ResolveRangeGate(SearchGate,SearchGate,TimeG,AngleG);
        {------------------------------}
        { Update MHR logic and settings }
        {------------------------------}
        if (TimeG[1]>0) OR (TimeG[2]>0) then
           begin
              ACQUIRE := TRUE;
              AcqGate := SearchGate;
              CalculateRangeError(AcqGate,AcqGate,TimeG,RError);
              CalculateAngleError(AngleG,AError);
              range     := range + RError;
              direction := direction + AError;
              if direction>2*pi then
                 begin
                    direction := direction - 2*pi;
                 end;
              if direction<0 then
                 begin
                    direction := direction + 2*pi;
                 end;
              InitRErrorStack;
              InitAErrorStack;
           end;
     end;
end;

procedure SendTrackPulse;
{-----------------------------------------------}
{ Procedure to simulate a MHR acquisition pulse }
{-----------------------------------------------}
var error   : real;
    TimeG,
    AngleG  : vectortype;
    RError,
    AError  : real;

begin
   {-------------------------------}
   { Set Range Gate for next pulse }
   {-------------------------------}
   if ACQUIRE then
      begin
         AcqGate   := AcqGate * AcqFactor;
         if AcqGate < 40.0 then
            begin
               AcqGate := 40.0;
            end;
      end
   else
      {----------------}
      { Tracking Phase }
      {----------------}
      begin
         AcqGate := 40.0;
      end;
```

```
{--------------}
{ Set RG width }
{--------------}
with SSM.gate do
   begin
      if range>sidelobeR then
         {-----------------------------------}
         { Outside effective side lobe range }
         {-----------------------------------}
         begin
            GateWidth := range * tan(radian(HalfBeam));
         end
      else
         {----------------------------------}
         { Inside effective side lobe range }
         {----------------------------------}
         begin
            GateWidth := range * tan(radian(3*HalfBeam));
         end;
   end;
{------------}
{ Send Pulse }
{------------}
ResolveRangeGate(AcqGate,AcqGate,TimeG,AngleG);
if (TimeG[1]>0) OR (TimeG[2]>0) then
   begin
      {--------------------------------}
      { Process range gate information }
      {--------------------------------}
      with SSM.gate do
         begin
            CalculateRangeError(AcqGate,AcqGate,TimeG,RError);
            range     := range + RError;
            CalculateAngleError(AngleG,AError);
            direction := direction + AError;
            if direction > 2*pi then
               begin
                  direction := direction - 2*pi;
               end;
            if direction < 0 then
               begin
                  direction := direction + 2*pi;
               end;
            {-----------------------------------------}
            { Once at minimum range gate, store errors }
            {-----------------------------------------}
            if AcqGate=40.0 then
               begin
                  PushRError(RError);
                  PushAError(AError);
               end;
            {----------------------------------------}
            { Once full stack, calculate range gate }
            { vectors and track                     }
            {----------------------------------------}
            if FullAStack  then
               begin
                  meanRvector := 0;
                  meanAvector := 0;
```

```
                    {--------------------------------}
                    { Mean range rate between pulses }
                    {--------------------------------}
                    while NOT(EmptyAStack) do
                       begin
                          meanAvector := meanAvector + PopAError;
                       end;
                    meanAvector := meanAvector*prf/stacksize;
                    {--------------------------------}
                    { Mean angle rate between pulses }
                    {--------------------------------}
                    while NOT(EmptyRStack) do
                       begin
                          meanRvector := meanRvector + PopRError;
                       end;
                    meanRvector := meanRvector*prf/stacksize;
                    {-------------------------------}
                    { If SSM in acquisition phase   }
                    { SSM advance to tracking phase }
                    {-------------------------------}
                    if ACQUIRE then
                       begin
                          LOCKED := TRUE;
                       end;
                    {----------------}
                    { Update Command }
                    {----------------}
                    UpdateAim := TRUE;
                    {------------}
                    { Set Stacks }
                    {------------}
                    InitAErrorStack;
                    InitRErrorStack;
                 end;
              end;
        end
     else
        begin
           SEARCH  := TRUE;
           if SSM.gate.range>MinRVal then
              begin
                 SetSearchGate(MaxSAngle,SSM.gate.range);
              end
           else
              begin
                 SetSearchGate(MaxSAngle,LSearchL);
              end;
           ACQUIRE := FALSE;
           LOCKED  := FALSE;
        end;
end;

procedure CalculateVector(dir, sp : real; var vector : vectortype);
{-----------------------------------------------------------}
{ Procedure to change polar vectors into Cartesian vectors }
{ Note : Bearing or Direction in radians                   }
{-----------------------------------------------------------}
begin
   vector[1] := sp*cos(dir);
   vector[2] := sp*sin(dir);
end;
```

```
procedure GetWindDirection(var X : real);
{----------------------------------------------}
{ Procedure to generate wind vector direction }
{----------------------------------------------}

var U,
    lower,
    upper     : real;
    found : boolean;
    J,K    : integer;

begin
   found := FALSE;
   U     := rand(WindDirectionStream);
   J     := 1;
   while not(found) do
      begin
         with wind do
            begin
               if direction[J]<U then
                  begin
                     J := J + 1;
                  end
               else
                  begin
                     found := TRUE;
                     J     := J - 1;
                     K     := J + 1;
                  end;
            end;
      end;
   lower := J * 10;
   upper := K * 10;
   with wind do
      begin
         X := lower + (upper - lower)
              *(U - direction[J])/(direction[K]-direction[J]);
      end;
end;

procedure GetWindSpeed(dir : real; var X : real);
{------------------------------------------}
{ Procedure to generate wind vector speed }
{------------------------------------------}
const test  = 22.5;
var found   : boolean;
    I       : integer;
    bearing,
    U,
    P       : real;
begin
   {--------------------------------}
   { Find octant for wind direction }
   {--------------------------------}
   found   := FALSE;
   bearing := 0;
   I       := 1;
   while not(found) do
      begin
         if abs(bearing - dir)<test then
            begin
               found := TRUE;
            end
```

```
         else
            begin
               bearing := bearing + 45;
               I := I + 1;
            end;
      end;
   if I = 9 then
      begin
         I := 1;
      end;
   {-----------------}
   { Find wind speed }
   {-----------------}
   U := rand(WindSpeedStream);
   P := 4 * U;
   J := trunc(P) + 1;
   with wind do
      begin
         X := speed[I,J] + (P - J + 1)*(speed[I,J+1]-speed[I,J]);
      end;
end;

procedure GenerateWindVector(var WindV : vectortype);
{---------------------------------------------------------}
{ Generate wind direction and speed as Cartesian vectors }
{---------------------------------------------------------}
var Wdirection,
    WdirRad,
    Wspeed       : real;
begin
   GetWindDirection(Wdirection);
   GetWindSpeed(Wdirection,Wspeed);
   Wdirection := CartForm(Wdirection);
   WdirRad  := radian(Wdirection);
   CalculateVector(WdirRad,Wspeed,WindV);
end;

procedure GenerateCurrent(var current : vectortype);
{---------------------------------------------------}
{ Procedure to model current as a Cartesian vector }
{---------------------------------------------------}
var v : vectortype;

begin
   {-------------------}
   { Current Direction }
   {-------------------}
   v[1] := uniform(210,250,CurrentDirectionStream);
   {---------------}
   { Current Speed }
   {---------------}
   v[2] := triang(currentA,currentB,currentC,CurrentVelocityStream);
   v[1] := radian(CartForm(v[1]));
   CalculateVector(v[1],v[2],current);
end;

procedure ShipRCSMag;
{---------------------------}
{ Set up ship RCS magnitudes }
{---------------------------}
var count     : integer;
```

```
begin
   for count := 1 to RCSPts do
      begin
         RCSship[i,3] := ship.rcsmag[RCSIndex,i];
      end;
   RCSTest := RCSIndex;
end;

procedure ShipRCSPlan;
{---------------------------}
{ Set up ship RCS positions }
{---------------------------}
var count    : integer;
    angle,
    distance : real;

begin
   for count:= 1 to RCSPts do
      begin
        {---------------------------------------}
        { Find bearing of RCS point from origin }
        {---------------------------------------}
         if ship.rcsposn[count,2]<>0 then
            begin
               if ship.rcsposn[count,1]>0 then
                  begin
                     angle := arctan(ship.rcsposn[count,2]/
                              ship.rcsposn[count,1]);
                  end
               else
                  begin
                     angle := pi + arctan(ship.rcsposn[count,2]/
                              ship.rcsposn[count,1]);
                  end;
            end
         else
            begin
               if ship.rcsposn[count,1]>0 then
                  begin
                     angle := 0;
                  end
               else
                  begin
                     angle := pi;
                  end;
            end;
         {-----------------------------------}
         { Find concurrent angle in the grid }
         {-----------------------------------}
         angle := angle + ship.head;
         if angle<0 then
            begin
               angle := angle + 2*pi;
            end;
         if angle>(2*pi) then
            begin
               angle := angle - 2*pi;
            end;
```

```
         {-------------------------------------}
         { Update ship's RCS reflective areas }
         {-------------------------------------}
         distance      := sqrt(sqr(ship.rcsposn[count,1])
                          + sqr(ship.rcsposn[count,2]));
         RCSship[count,1] := ship.posn[1] + distance*cos(angle);
         RCSship[count,2] := ship.posn[2] + distance*sin(angle);
      end;
   {-------------------------}
   { Ship's Centre position }
   {-------------------------}
   angle := ship.head + pi;
   if angle<0 then
      begin
         angle := angle + 2*pi;
      end;
   if angle>(2*pi) then
      begin
         angle := angle - 2*pi;
      end;
   ShipCentre[1] := ship.posn[1] + 10*cos(angle);
   ShipCentre[2] := ship.posn[2] + 10*sin(angle);
end;

procedure SSMAlterCourse;
{---------------------------------------------}
{ Procedure to increment SSM flight direction }
{---------------------------------------------}
var PursuitAngle : real;
    PURDIRRIGHT  : boolean;
begin
   PursuitAngle := SSM.aim - SSM.head;
   if PursuitAngle < - pi then
      begin
         PursuitAngle := PursuitAngle + 2*pi;
      end;
   if PursuitAngle > pi then
      begin
         PursuitAngle := -(2*pi - PursuitAngle);
      end;
   if PursuitAngle > 0 then
      begin
         PURDIRRIGHT := FALSE;
      end
   else
      begin
         PURDIRRIGHT := TRUE;
      end;
   {---------------------------------}
   { Increment SSM flight direction }
   {---------------------------------}
   if (abs(PursuitAngle))>(radian(MTR)/prf) then
      begin
         if PURDIRRIGHT then
            begin
               SSM.head := SSM.head - radian(MTR/prf);
            end
         else
            begin
               SSM.head := SSM.head + radian(MTR/prf);
            end;
      end
```

```
   else
      begin
         SSM.head := SSM.aim;
      end;
   CalculateVector(SSM.head,SSM.speed,SSM.move);
end;

procedure ShipDirectionFind (SSMpos : vectortype;
                             var shipdir : real);
{--------------------------------------------}
{ Procedure to find ship's direction from SSM }
{--------------------------------------------}
begin
   if SSM.posn[1]>0 then
      begin
         shipdir := pi + arctan(ssm.posn[2]/ssm.posn[1]);
      end
   else
      begin
         shipdir := arctan(ssm.posn[2]/ssm.posn[1]);
         if shipdir<0 then
            begin
               shipdir := shipdir + 2*pi;
            end;
      end;
end;

procedure SSMModel;
{----------------------------------}
{ Procedure to simulate SSM object }
{----------------------------------}
var TimeGain,
    jamdir     : real;

begin
   {-----------------------------}
   { Increment missile's position }
   {-----------------------------}
   SSM.posn[1] := SSM.posn[1] + (SSM.move[1] + WindV[1])/prf;
   SSM.posn[2] := SSM.posn[2] + (SSM.move[2] + WindV[2])/prf;
   {-----------------------------------------------}
   { Assume ballistic flight path for final 150 meters }
   {-----------------------------------------------}
   if NOT(BALLISTIC) and (SSM.gate.range<150.0) then
      begin
         BALLISTIC := TRUE;
      end;
   {-------------------------}
   { Activate MHR if required }
   {-------------------------}
   if NOT(MHR)AND
     (sqrt(sqr(SSM.posn[1])+sqr(SSM.posn[2]))<MHRactRange) then
      begin
         MHR    := TRUE;
         SEARCH := TRUE;
         SetSearchGate(MaxSAngle,LSearchL);
      end;
   {--------------------------}
   { Home on Jam when required }
   {--------------------------}
   if JAMMER AND NOT(HOJ) then
      begin
         HOJ := TRUE;
      end;
```

```
if HOJ AND NOT(JAMMER) then
   begin
      HOJ := FALSE;
   end;
{---------------------}
{ Search if required }
{---------------------}
if SEARCH then
   begin
      if NOT(HOJ) then
         begin
            SendSearchPulse;
         end
      else
         begin
            ShipDirectionFind(SSM.posn,jamdir);
            SSM.gate.direction := jamdir;
            SSM.aim            := jamdir;
            if NOT(PURSUIT) then
               begin
                  PURSUIT := TRUE;
               end;
         end;
   end;
{----------------------}
{ Acquire if required }
{----------------------}
if ACQUIRE AND NOT(BALLISTIC) then
   begin
      if SEARCH then
         begin
            SEARCH := FALSE;
            PURSUIT := TRUE;
         end
      else
         begin
            if NOT(HOJ) then
               begin
                  SendTrackPulse;
               end
            else
               begin
                  ShipDirectionFind(SSM.posn,jamdir);
                  SSM.gate.direction := jamdir;
                  SSM.aim            := jamdir;
               end;
         end;
   end;
{--------------------}
{ Track if required }
{--------------------}
if LOCKED AND NOT(BALLISTIC) then
   begin
      if ACQUIRE then
         begin
            ACQUIRE   := FALSE;
            PURSUIT   := FALSE;
            LEADANGLE := TRUE;
         end
      else
         begin
            if NOT(HOJ) then
               begin
                  SendTrackPulse;
               end
```

```
              else
                 begin
                    ShipDirectionFind(SSM.posn,jamdir);
                    SSM.gate.direction := jamdir;
                    SSM.aim            := jamdir;
                    SSM.gate.range     := SSM.gate.range +
                                          MeanRVector/prf;
                    if LEADANGLE then
                       begin
                          PURSUIT      := TRUE;
                          LEADANGLE    := FALSE;
                       end;
                 end;
           end;
     end;
   {----------------}
   { SSM Manoeuvre }
   {-----------------}
   { SSM in pursuit }
   {-----------------}
   if PURSUIT AND NOT(SEARCH) then
      begin
         {------------------------------}
         { Set aiming direction for SSM }
         {------------------------------}
         SSM.aim := SSM.gate.direction;
         {----------------------------}
         { Alter SMM direction vector }
         {----------------------------}
         SSMAlterCourse;
      end;
   {--------------------------------}
   { SSM in proportional navigation }
   {--------------------------------}
   if LEADANGLE AND NOT(BALLISTIC)  then
      begin
         if UpdateAim then
            begin
               TimeGain   := SSM.gate.range/abs(MeanRVector);
               SSM.aim    := SSM.aim + LAGain*TimeGain*MeanAVector;
               UpdateAim := FALSE;
               SSMAlterCourse;
            end
         else
            begin
               SSMAlterCourse;
            end;
      end;
end;

procedure ExecuteTactics;
{------------------------------------------------------------}
{ Procedure to enable tactics to be executed by ship model }
{------------------------------------------------------------}
var SSMbearing,
    SSMrange,
    DiffBrg      : real;
    index        : integer;

begin
   {--------------------}
   { Find missile range }
   {--------------------}
   SSMrange := RangeTest;
```

```
{----------------------}
{ Find missile bearing }
{----------------------}
if SSM.posn[1] > 0 then
   begin
      SSMbearing := arctan(SSM.posn[2]/SSM.posn[1]);
      if SSMbearing < 0 then
         begin
            SSMbearing := SSMbearing + 2*pi;
         end;
   end
else
   begin
      SSMbearing := pi + arctan(SSM.posn[2]/SSM.posn[1]);
   end;
if Tactic_A then
   {---------------------------}
   { Increase speed to maximum }
   {---------------------------}
   begin
      {--------------------------------------}
      { Set up commencement time for tactic }
      {--------------------------------------}
      TactABTime := STime + 3.0 + weibull(1.75,TactABDelayStream);
      {-------------------------------------------}
      { Set maximum speed and speed change logic }
      {-------------------------------------------}
      NewSpeed   := 0.512 * MaxSpeed;
      ASpeed     := TRUE;
   end;
if Tactic_B then
   {------------------------------------------------}
   { Alter course perpendicular to missile bearing }
   {------------------------------------------------}
   begin
      {--------------------------------------}
      { Set up commencement time for tactic }
      {--------------------------------------}
      if TactABTime = InitTime then
         begin
            TactABTime := STime + 3.0 +
                          weibull(1.75,TactABDelayStream);
         end;
      {--------------------------------}
      { Determine direction of the turn }
      {--------------------------------}
      if ((SSMbearing>radian(270))AND(ship.head<radian(90)))OR
         ((SSMbearing<radian(90))AND(ship.head>radian(270))) then
         begin
            if SSMbearing>radian(270) then
               begin
                  DiffBrg := 2*pi - SSMbearing + ship.head;
               end
            else
               begin
                  DiffBrg := -(2*pi - ship.head + SSMbearing);
               end;
         end
```

```
   else
      begin
         DiffBrg := ship.head - SSMbearing;
         if abs(DiffBrg) > pi then
            begin
               if DiffBrg < 0 then
                  begin
                     DiffBrg := 2*pi + DiffBrg;
                  end
               else
                  begin
                     DiffBrg := DiffBrg - 2*pi;
                  end;
               end;
      end;
   if abs(DiffBrg) < radian(90) then
      begin
         if DiffBrg < 0 then
            begin
               TurnPort := FALSE;
            end
         else
            begin
               TurnPort := TRUE;
            end;
      end
   else
      begin
         if DiffBrg < 0 then
            begin
               TurnPort := TRUE;
            end
         else
            begin
               TurnPort := FALSE;
            end;
      end;
   {-----------------------}
   { Determine new heading }
   {-----------------------}
   if DiffBrg < 0 then
      begin
         NewCourse := SSMbearing - radian(90);
      end
   else
      begin
         NewCourse := SSMbearing + radian(90);
      end;
   if NewCourse > 2*pi then
      begin
         NewCourse := NewCourse - 2*pi;
      end;
   if NewCourse < 0 then
      begin
         NewCourse := NewCourse + 2*pi;
      end;
   NewCourse := radian(trunc(degree(NewCourse)));
   {-------------------}
   { Set turning logic }
   {-------------------}
   ACourse := TRUE;
end;
```

```
   if Tactic_C then
      {----------------------------------}
      { Jam to obscure chaff deployment }
      {----------------------------------}
      begin
         JAM := TRUE;
      end;
   if Tactic_D then
      {--------------------------}
      { Deploy close range chaff }
      {--------------------------}
      begin
         {----------------------------}
         { Determine CRC firing times }
         {----------------------------}
         if SSMrange < CRCdepR then
            begin
               CRCfTime[1] := STime + weibull(1.5,TactDRangeStream);
            end
         else
            begin
               CRCfTime[1] := STime + (SSMRange - CRCdepR)/SSMestSp;
            end;
         for index := 2 to maxCRC do
            begin
               CRCfTime[index] := CRCfTime[index-1] + CRCint;
            end;
         {---------------}
         { Set CRC logic }
         {---------------}
         CRCdeploy := TRUE;
         CRCindex  := 1;
      end;
   if Tactic_E then
      {---------------------------}
      { Deploy medium range chaff }
      {---------------------------}
      begin
         if SSMrange < MRCdepR then
            begin
               MRCfTime := STime + weibull(1.5,TactERangeStream);
            end
         else
            begin
               MRCfTime := STime + (SSMRange - MRCdepR)/SSMestSp;
            end;
         MRCdeploy := TRUE;
      end;
   TacticOrdered := TRUE;
end;

procedure ShipModel;
{--------------------------------------}
{ Procedure to simulate FAC(M) object }
{--------------------------------------}
var SSMdist : real;
    chaff_i : integer;
```

```
begin
   {-----------------------------------------}
   { Scan for missile presence and when detected }
   { initialise ordered tactics                }
   {-----------------------------------------}
   SSMdist := sqrt(sqr(SSM.posn[1])+sqr(SSM.posn[2]));
   if not(TacticOrdered) then
      begin
         if (SSMdist < RSR) OR (MHR AND (SSMdist < MHRDetect)) then
            begin
               ExecuteTactics;
            end;
      end;
   {------------------------------------}
   { Enable Tactics at appropriate time }
   {------------------------------------}
   if not(ReactionTimeElapsed) AND (Stime > TactABTime) then
      begin
         ReactionTimeElapsed := TRUE;
      end;
   {---------------------------}
   { Increment ship's position }
   {-----------------------------------------------------}
   { If required by tactic, change ship movement vector }
   {-----------------------------------------------------}
   if (Acourse OR Aspeed) AND ReactionTimeElapsed then
      begin
         if abs(NewSpeed-ship.speed)>SpeedDiff then
            begin
               ship.speed := ship.speed + SRate/prf;
            end
         else
            begin
               Aspeed     := FALSE;
               ship.speed := NewSpeed;
            end;
         if abs(NewCourse-ship.head)>CourseDiff then
            begin
               if TurnPort then
                  begin
                     ship.head := ship.head + turnrate/prf;
                     if ship.head>(2*pi) then
                        begin
                           ship.head := ship.head - 2*pi;
                        end;
                  end
               else
                  begin
                     ship.head := ship.head - turnrate/prf;
                     if ship.head<0 then
                        begin
                           ship.head := ship.head + 2*pi;
                        end;
                  end;
            end
         else
            begin
               Acourse   := FALSE;
               ship.head := NewCourse;
            end;
         CalculateVector(ship.head,ship.speed,ship.move);
         ShipRCSPlan;
      end;
```

```
{--------------------------------------------------}
{ Calculate new ship's position in regard to SSM }
{--------------------------------------------------}
{ Pivoting Point }
{----------------}
SSM.posn[1] := SSM.posn[1] - (ship.move[1] + CurrentV[1])/prf;
SSM.posn[2] := SSM.posn[2] - (ship.move[2] + CurrentV[2])/prf;
{---------------------------------------}
{ Update ship's radar reflective areas }
{---------------------------------------}
SSMRelBrg(SSMBrg,degree(ship.head),RCSIndex);
if RCSIndex<>RCSTest then
   begin
      ShipRCSMag;
   end;
{----------------------------}
{ Increment chaff positions }
{----------------------------}
if ChaffCount > 0 then
   begin
      for chaff_i := 1 to ChaffCount do
         begin
            chafflist[chaff_i,1] := chafflist[chaff_i,1] +
                                    (WindV[1] - ship.move[1] -
                                     CurrentV[1])/prf;
            chafflist[chaff_i,2] := chafflist[chaff_i,2] +
                                    (WindV[2] - ship.move[2] -
                                     CurrentV[2])/prf;
         end;
   end;
{--------------------------------------------}
{ Fire close range chaff at appointed time }
{--------------------------------------------}
if CRCdeploy AND (STime>CRCfTime[CRCindex]) then
   begin
      FireCRC;
      CRCindex := CRCindex + 1;
      if CRCindex > maxCRC then
         begin
            CRCdeploy := FALSE;
         end;
   end;
{---------------------------------------------}
{ Fire medium range chaff at appointed time }
{---------------------------------------------}
if MRCdeploy AND (STime>MRCfTime) then
   begin
      FireMRC;
      MRCdeploy := FALSE;
   end;
{----------------}
{ Operate Jammer }
{----------------}
if JAM then
   begin
     if NOT(JAMMER) then
        begin
           JAMMER := TRUE;
           JTime  := STime + JamTime;
        end
```

```
         else
            begin
               if Tactic_D then
                  begin
                     if CRCfTime[maxCRC]<Stime then
                        begin
                           JAM    := FALSE;
                           JAMMER := FALSE;
                        end;
                  end
               else
                  begin
                     if JTime<Stime then
                        begin
                           JAM    := FALSE;
                           JAMMER := FALSE;
                        end;
                  end;
            end;
      end;
end;

procedure InitialiseGlobalVariablesReplication;
{---------------------------------------------------------------}
{ Initialise Global Variables for a particular Simulation run }
{---------------------------------------------------------------}
begin
   {-------------------------------}
   { End of simulation replication }
   {-------------------------------}
   EOS := FALSE;
   {------------------}
   { Reset Clock Time }
   {------------------}
   Stime := 0;
   {------------------------}
   { Initialise Error Stacks }
   {------------------------}
   InitRErrorStack;
   InitAErrorStack;
   {------}
   { Wind }
   {------}
   GenerateWindVector(WindV);
   {---------}
   { Current }
   {---------}
   GenerateCurrent(CurrentV);
   {-------------}
   { Temperature }
   {-------------}
   temp := normal(tempM,tempSD,TemperatureStream);
   {-----------------------------------------}
   { Ship initial position, course and speed }
   {-----------------------------------------}
   with ship do
      begin
         posn[1]    := 0;
         posn[2]    := 0;
         head       := radian(trunc(uniform(0,360,ShipsHeadStream)
                                  + 0.5));
         speed      := 0.512 * trunc(uniform(22,28,ShipsSpeedStream)
                                  + 0.5);
         NewSpeed   := speed;
         NewCourse  := head;
```

```
      CalculateVector(head,speed,move);
   end;
{---------------------------}
{ Set up initial ship logic }
{---------------------------}
ACourse              := FALSE;
ASpeed               := FALSE;
MHR                  := FALSE;
JAM                  := FALSE;
JAMMER               := FALSE;
SRdetSSM             := FALSE;
ReactionTimeElapsed  := FALSE;
TurnPort             := FALSE;
CRCPort              := FALSE;
TacticOrdered        := FALSE;
CRCdeploy            := FALSE;
MRCdeploy            := FALSE;
{-------------------------------------------}
{ Set up Search Radar detection range on SSM }
{-------------------------------------------}
RSR      := DetectionRange(SR)*1000;
{---------------------------------}
{ Set up initial missile position }
{---------------------------------}
with SSM do
   begin
      SSMBrg      := uniform(0,360,SSMbrgStream);
      SSMRange    := SSMminR + uniform(SSMminR,SSMmaxR,
                                       SSMRngStream);
      SSMRange    := SSMRange * 1843.2;
      CalculateVector(radian(SSMBrg),SSMRange,posn);
      {-------------------------------}
      { Set up initial missile course }
      {-------------------------------}
      head   := SSMBrg + 180;
      head   := normal(head,SSMhSD,SSMHeadingStream);
      while head>360 do
         begin
            head := head - 360;
         end;
      head   := radian(head);
      {----------------------}
      { Set up missile speed }
      {----------------------}
      speed := Mach * sqrt(y*R*(273+temp));
      CalculateVector(head,speed,move);
   end;
{------------------------------}
{ Set up initial missile logic }
{------------------------------}
SEARCH     := FALSE;
ACQUIRE    := FALSE;
LOCKED     := FALSE;
BALLISTIC  := FALSE;
HOJ        := FALSE;
PURSUIT    := FALSE;
LEADANGLE  := FALSE;
{-----------------------------------------------------------}
{ Set up missile gate range to facilitate ballistic logic }
{-----------------------------------------------------------}
SSM.gate.range := USearchL;
```

```
   {-------------------------------------------}
   { Set relative SSM relative bearing to ship }
   {-------------------------------------------}
   SSMRelBrg(SSMBrg,degree(ship.head),RCSIndex);
   RCSTest := 40;
   {--------------------------------}
   { Set ship's RCS characteristics }
   {--------------------------------}
   ShipRCSPlan;
   ShipRCSMag;
   {----------------------------}
   { Set up MHR detection range }
   {----------------------------}
   MHRDetect :=min(normal(ESMAdv*RSR,sqrt(RSR/2),MHRDetectStream),
                   RHorizon);
   {-----------------}
   { Initialise Chaff }
   {-----------------}
   for i:= 1 to 80 do
      begin
         for j:= 1 to 3 do
            begin
               chafflist[i,j] := 0;
            end;
      end;
   {-------------------------------}
   { Number of Chaff rockets fired }
   {-------------------------------}
   ChaffNumber := 0;
   {----------------------------------}
   { Number of chaff bundles deployed }
   {----------------------------------}
   ChaffCount  := 0;
   {----------------------}
   { Initialise range test }
   {----------------------}
   RangeTest   := 9999999999.0;
   {-----------------------}
   { Initialise delay times }
   {-----------------------}
   TactABTime  := InitTime;
   TactCDETime := InitTime;
   write(reportfile,'REPLICATION :',counter:6);
   clrscr;
   gotoXY(10,10);
   write('Replication Number :',counter:10,' ...');
end;

procedure InitialiseGlobalVariablesSimulation;
{------------------------------------------------------}
{ Initialise Global Variables for all Simulation Runs }
{------------------------------------------------------}
begin
   clrscr;
   {--------------------------------------------}
   { Set up random number seeds for all streams }
   {--------------------------------------------}
   Randdf;
   {----------------------------}
   { Set up simulation time step }
   {----------------------------}
   Dtime := 1/prf;
```

```
{---------------------------------------}
{ Set up streams for random number rows }
{---------------------------------------}
WindDirectionStream    :=  1;
WindSpeedStream        :=  2;
CurrentDirectionStream :=  3;
CurrentVelocityStream  :=  4;
TemperatureStream      :=  5;
ShipsHeadStream        :=  6;
ShipsSpeedStream       :=  7;
SRDetectStream         :=  8;
SSMBrgStream           :=  9;
SSMRngStream           := 10;
SSMHeadingStream       := 11;
DetectRadarStream      := 12;
MHRDetectStream        := 13;
ESMbrgStream           := 14;
CRCStream              := 15;
MRCStream              := 16;
TactABDelayStream      := 17;
TactCDelayStream       := 18;
TactDRangeStream       := 19;
TactERangeStream       := 20;
{-------------------------}
{ Set up wind parameters }
{-------------------------}
assign(windfile,'wind.dat');
reset(windfile);
read(windfile,wind);
close(windfile);
{---------------------------------------------}
{ Set up Search Radar detection distribution }
{---------------------------------------------}
assign(RadarDetectFile,'RADAR.DAT');
reset(RadarDetectFile);
for i:= 1 to 5 do
   begin
      read(RadarDetectFile,SR[i]);
   end;
close(RadarDetectFile);
{----------------------}
{ Set up ship RCS data }
{----------------------}
{ Positional RCS data}
{--------------------}
assign(rcsposfile,'rcspos.dat');
reset(rcsposfile);
for i := 1 to RCSPts do
   begin
      for j := 1 to 2 do
         begin
            with ship do
               begin
                  read(rcsposfile,rcsposn[i,j]);
               end;
         end;
   end;
close(rcsposfile);
```

```
   {--------------------}
   { RCS strength data }
   {--------------------}
   assign(rcsmagfile,'rcsmag.dat');
   reset(rcsmagfile);
   for i := 0 to RelDir do
      begin
         for j := 1 to RCSPts do
            begin
               read(rcsmagfile,ship.rcsmag[i,j]);
            end;
      end;
   close(rcsmagfile);
   {--------------------------}
   { Set up turning rate data }
   {--------------------------}
   assign(CRfile,'altco.dat');
   reset(CRfile);
   for i := 11 to 15 do
      begin
         for j := 1 to 9 do
            begin
               read(CRfile,CRate[i,j]);
            end;
      end;
   close(CRfile);
   {-------------------------}
   { Initialise missile hits }
   {-------------------------}
   Mhit := 0;
   {-----------------------------------}
   { Initialise missile miss distance }
   {-----------------------------------}
   MissDistance := 0;
end;

procedure VerifyEOS;
{-----------------------}
{ Check SSM hit criteria }
{-----------------------}
var dist : real;

begin
   dist := sqrt(sqr(SSM.posn[1]-ShipCentre[1])+
                sqr(SSM.posn[2]-ShipCentre[2]));
   if dist > RangeTest then
      begin
         if dist < hitdist then
            begin
               Mhit := Mhit + 1;
               writeln(reportfile,'  HIT      ',RangeTest:10:2);
            end
         else
            begin
               writeln(reportfile,'  MISSED  ',RangeTest:10:2);
            end;
         EOS := TRUE;
         MissDistance := MissDistance + RangeTest;
      end
   else
      begin
         RangeTest := dist;
      end;
end;
```

```
procedure FinalReport;
{-----------------------------------------------}
{ Procedure to present final report on screen }
{-----------------------------------------------}
begin
   {------------------------}
   { Final report on screen }
   {------------------------}
   clrscr;
   writeln;
   writeln('   TACTICS EMPLOYED');
   writeln;
   if Tactic_A then
      begin
         writeln('   +  Increased speed to 30 knots');
      end
   else
      begin
         writeln('   -  Speed remain constant');
      end;
   if Tactic_B then
      begin
         writeln('   +  Alter course to maximise SSM angular rate');
      end
   else
      begin
         writeln('   -  Course remain constant');
      end;
   if Tactic_C then
      begin
         writeln('   +  Jam MHR to obscure chaff deployment');
      end
   else
      begin
         writeln('   -  No Jamming');
      end;
   if Tactic_D then
      begin
         writeln('   +  Deployed Short Range Chaff');
      end
   else
      begin
         writeln('   -  Short Range Chaff not deployed');
      end;
   if Tactic_E then
      begin
         writeln('   +  Deployed Medium Range Chaff')
      end
   else
      begin
         writeln('   -  Medium Range Chaff not deployed');
      end;
   writeln;
   writeln('   SIMULATION RESULTS');
   writeln;
   writeln('   Hits Obtained      : ',Mhit:5,' out of ',maxrun:5);
   writeln('   Mean Miss Distance : ',MissDistance:10:2,' meters');
```

```
  {----------------------}
  { Final report to file }
  {----------------------}
  writeln(reportfile);
  writeln(reportfile,'TACTICS EMPLOYED');
  writeln(reportfile);
  if Tactic_A then
     begin
        writeln(reportfile,'+  Increased speed to 30 knots');
     end
  else
     begin
        writeln(reportfile,'-  Speed remain constant');
     end;
  if Tactic_B then
     begin
        writeln(reportfile,'+  Alter course to maximise SSM',
                ' angular rate');
     end
  else
     begin
        writeln(reportfile,'-  Course remain constant');
     end;
  if Tactic_C then
     begin
        writeln(reportfile,'+  Jam MHR to obscure chaff',
                ' deployment');
     end
  else
     begin
        writeln(reportfile,'-  No Jamming');
     end;
  if Tactic_D then
     begin
        writeln(reportfile,'+  Deployed Short Range Chaff');
     end
  else
     begin
        writeln(reportfile,'-  Short Range Chaff not deployed');
     end;
  if Tactic_E then
     begin
        writeln(reportfile,'+  Deployed Medium Range Chaff')
     end
  else
     begin
        writeln(reportfile,'-  Medium Range Chaff not deployed');
     end;
  writeln(reportfile);
  writeln(reportfile,'SIMULATION RESULTS');
  writeln(reportfile);
  writeln(reportfile,'   Hits Obtained      : ',Mhit:5,
                     ' out of ',maxrun:5);
  writeln(reportfile,'   Mean Miss Distance : ',MissDistance:10:2,
                     ' meters');
  {-------------------------------}
  { Termination message to screen }
  {-------------------------------}
  gotoXY(5,20);
  write('<< Press any Key to Terminate Simulation >>');
  repeat until keypressed;
end;
```

```
begin {Main}
   {---------------------}
   { Prepare Report File }
   {---------------------}
   assign(reportfile,'report.txt');
   rewrite(reportfile);
   InitialiseGlobalVariablesSimulation;
   {-----------------------------------------------}
   { Do ordered number of simulation replications }
   {-----------------------------------------------}
   for counter := 1 to maxrun do
      begin
         {----------------------------------}
         { Do single simulation replication }
         {----------------------------------}
         InitialiseGlobalVariablesReplication;
         while NOT(EOS) do
            begin
               {--------------------------}
               { Increment simulation time }
               {--------------------------}
               Stime := Stime + Dtime;
               {----------------------}
               { Increment Ship model }
               {----------------------}
               ShipModel;
               {---------------------}
               { Increment SSM model }
               {---------------------}
               SSMModel;
               {------------------------------}
               { Check for end of replication }
               {------------------------------}
               VerifyEOS;
            end;
         {----------------------------}
         { Report Simulation Findings }
         {----------------------------}
      end;
   MissDistance := MissDistance/maxrun;
   FinalReport;
   close(reportfile);
end.  {Main}
```

Appendix H

PRACTICAL PROGRAMMING LESSONS LEARNT

This Appendix lists five main problems that were encountered whilst programming the computer model. Although most programmers might be conversant with these types of problems, they are repeated here to show that it might be prudent to keep them in mind when programming similar constructs or modules.

H.1 HANDLING THE SQUARE ROOT OF LARGE NUMBERS

Whilst implementing an early radar detection model of the form

$$R_{\text{detect}} = p_{\text{detect}} R_{\text{max}} = p_{\text{detect}} \sqrt[4]{\frac{P_t G^2 \lambda^2 \sigma}{(4\pi)^3 S_{\text{min}} L}}$$

where p_{detect} is a factor with which R_{max} is multiplied to generate some detection range which is less than the very favourable R_{max}, the fourth root portion of the right hand term was written in Pascal as

```
sqrt(sqrt((P*sqr(G)*sqr(lambda)*sigma)/(power((4*pi),3)*minS*L))).
```

Note that the `power` function is inherent to the program. When checking the Pascal syntax and compiling the program, no errors were reported. However, when the program was executed, the program was terminated prematurely with an error message that the program attempted to calculate the square root of a negative number.

This message was difficult to understand as all the variables in the equation were positive numbers. Thus, the multiplication of all the variables should have resulted in another positive number. After much deliberation, it was found that, because the variables were large, the multiplication of all the variables led to a floating point under-flow which were interpreted by the compiler as a negative number. The problem was resolved by using Pascal's `ln`-function to evaluate the term.

Lesson Learnt. When large numbers may be encountered, care must be taken to use suitable methods or data types to deal with them.

H.2 SEARCH ALGORITHM

Whilst developing the search algorithm, it was found that, at one stage, the direction of the range gate would consistently oscillate around the edge of the search area. The logic of the search algorithm is to search in a particular direction until the limit on that side of the missile's heading was reached, then the direction would be changed for the missile head radar to search in the opposite direction.

In order to test whether the angular limit of the search area has been reached, the program finds the absolute value of the difference, d, between the missile's heading and the present missile head radar line-of-sight. Let the angular limit on either side of the missile bearing be l. If $d \geq l$, then the direction of the search would change.

After investigation of the problem, it was found that the procedure that set up the missile head line-of-sight prior to the missile operation assuming the search phase, was at fault. It set up the missile head radar line-of-sight on the left limit and ordered a right command. At first glance this logic seemed reasonable, but it was found that, when the search phase was ordered, the program detected that $d \geq l$ and, as a result, changed the search direction to the left. The line-of-sight of the next pulse then remained such that $d \geq l$. Again, the search direction was changed, but $d \geq l$ still held true. The result was that the missile head radar line-of-sight oscillated around the edge of the search area.

In order to resolve the problem, two methods were available. On the one hand, the search direction ordered in the procedure that set up the missile head line-of-sight prior to the missile operation assuming the search phase could simply be changed to be a left command. On ordering the search phase, the program would detect that $d \geq l$ and, as a result, changed the search direction to the right. On the other hand, the initial line-of sight could be placed well within the angular search limits. Thus, the program would have found $d < l$ and the search would be executed correctly.

After careful consideration, taking into account the fact that the search algorithm must also cater for the re-acquisition of the target in the event that it was lost by the missile head radar, the latter method was decided upon and the initial missile head radar line-of-sight was chosen to be $l/2$ on the left hand side of the missile heading.

Lesson Learnt. Care must be taken to initialise settings in such a manner that it does not compromise program logic.

H.3 RESOLVING THE RANGE GATE

In order to find the amount of energy in the range gate on a particular pulse, the positions of all the radar reflective points must first be transformed from the simulation Cartesian grid to the range gate co-ordinate system. In order to do this, the bearing and distance of all the radar reflective points were calculated in the Cartesian grid co-ordinate system. The result was a polar vector for every radar reflective point. In turn, this polar vector would be modified by the range gate direction to deliver a polar vector in the range gate co-ordinate system. Finally, this new polar vector was transformed in a Cartesian vector.

In order to ascertain whether a particular radar reflective point was in the range gate, it was necessary only to ascertain whether the radar reflective point was within the limits of the range gate. In other words, we tested to see whether the position of the radar reflective point was less than or equal to the values of the limits of the range gate.

A simple mistake was made in coding this test. Instead of using the origin of the range gate co-ordinate system, the origin of the range gate was coded in the simulation Cartesian co-ordinate system. This problem was extremely difficult to resolve as we chose the position of the ship to be the position (0,0) in the simulation Cartesian grid. Thus, as the value of the range gate co-ordinate system's origin was very close to (0,0) we often found seemingly good results. The problem was eventually resolved and the necessary corrective action was taken.

Lesson Learnt. When using more than one co-ordinate system within a program, it might be useful to choose variable names such that it is clear which co-ordinate system a particular variable refers to. For example, positions in the Cartesian and the range gate grids might be distinguished by using the variable names `CARTPOS` and `GATEPOS` respectively.

H.4 RUNNING A CONTINUOUS SIMULATION WITH DISCRETE EVENTS IN PARALLEL

The simulation is essentially continuous. This is achieved by incrementing the time in small fixed steps. However, several discrete events happen during the course of the simulation. An example is the case where the ship must commence altering course at a random discrete time. None of the available sources made any suggestions in this regard. We had to develop this method on our own. The method we developed for running a continuous simulation with discrete events in parallel was discussed in Section 6.5. For completeness' sake, it is repeated directly below.

"The reason for keeping track of the time elapsed is to enable the execution of discrete events in parallel with the continuous simulation of the FAC(M) and SSM movements and other inherent actions.

For example, if a change of speed will be ordered, it would be a discrete event that is dependent on whether the missile has been detected and the FAC(M) crew's reaction time which can be simulated by a stochastic process. We denote the replication time elapsed by T. Let us assume that the missile is detected at time T_i and that the crew's reaction time is t_r. The time at which the discrete event, that is, the beginning of the speed change, must happen will be at time $T_j = T_i + t_r$. Now, by testing at subsequent iterations of the ship model whether the time specified for the discrete event is equal or less than the simulation time, that is, whether $T \geq T_j$, the discrete event can be inserted at the appropriate time.

As the time increments in our model is sufficiently small, that is, the time increment is 0.002 seconds, any errors resulting from the situation where $T > T_j$ will be in the interval (0,0.002) seconds and can, in regard to this simulation model, be regarded as inconsequential."

Lesson Learnt. When dealing with both continuous and discrete event simulation in one program, the time elapsed since the start of the simulation replication will aid the implementation of both simultaneously.

H.5 REDUCING COMPUTER RUN TIME

In Section 7.1, we stated that the run time, t_{run}, was mainly dependent on the pulse repetition frequency, f. For the particular personal computer that was used to run the program, an experiment with one hundred replications took $t_{run} \approx f / 10$ minutes to complete the simulation. The main contributing factors to the run time were the number of radar reflective points whose position had to be incremented for every time step, Δt, and the resolution of the range gate for every pulse that was simulated. The latter contributing factor was discussed in Section H.3 above. As t_{run} behaved approximately in a linear manner with respect to f, we conclude to say that the program's time complexity is of the order $O(f)$.

Consider the number of radar reflective points whose positions had to be incremented for every time step. They constituted the missile's position, the seventeen radar reflective points that made up the FAC(M) and three bundle positions for every chaff rocket fired. If we consider the case where we chose to fire both close range chaff and medium range chaff, we see that we would have 30 bundle positions to increment at every time step. If we add the missile and the seventeen radar reflective points that made up the FAC(M) then we have to increment a total of 48 positions at every time step.

Type of Simulation	*Percentage Savings*
No close or medium range chaff deployed	88.9
Close range chaff deployed only	66.7
Medium range chaff deployed only	38.1
Both close and medium range chaff deployed	33.3

Table H.1: Savings on computer actions to effect incrementing the positions of all the radar reflective points.

By fixing the ship's turning point in position (0,0) in the Cartesian grid and incrementing the missile and the chaff positions relative to the ship's position, we could reduce the number of computer instructions to be carried out. In order to ensure that the missile moves relative to the ship, we subtracted the ship's course and speed vector from the missile's position at very time step in the ship model. Thus we have saved 16 of the total number of computer actions to increment the positions of all the radar reflective points. Table H.1 depicts the percentage of savings for various experiments.

Note that, if the radar cross section of the ship is implemented as a more dense model, the savings realised can be much bigger. Also, the percentage savings reported did not take into account the fact that, whilst the ship is turning, the seventeen associated radar reflective points must be updated around the centre position (0,0).

Lesson Learnt. Always bear in mind that you must endeavour to reduce the number of program step instructions in order to reduce the run time of the program. This is of particular importance when the program's time complexity is of an exponential or other order that behaves badly.

Appendix I

ALPHABETICAL INDEX

-T-

-V-

-W-

-Y-

Appendix J

SOURCES CONSULTED

Africa Pilot Volume III, South and East Coasts of Africa from Cape Agulhus to Ras Binnah, including the Islands of Zanzibar and Pemba. 13 ed., Taunton: Hydrographer of the Navy, 1980.

Bell, D., Morrey, I. and Pugh. J., *Software Engineering - A Programming Approach.* London: Prentice Hall, 1987.

Botha, P.L., *Toepassing van Radar Absorberende Materiaal vir Vermindering van die Aanvalsvaartuig Radardeursnit.* Simon's Town: IMT, October 1986.

Box, E.P., Hunter, W.G. and Hunter, J.S., *Statistics for Experimenters.* New York: John Wiley, 1978.

Bratley, P., Fox, B.L. and Schrage, L.E., *A Guide to Simulation.* 2 ed., New York: Springer, 1987.

Burger, M., *Reference Techniques.* 8 rev. Pretoria: UNISA, 1992.

Clemen R.T., *Making Hard Decisions : An Introduction to Decision Analysis*, Boston: PWS-Kent, c1991.

Couper, D.J., *Simulasie*. Pretoria: Universiteit van Suid-Afrika, 1985.

Dale, N. and Lilly, S., *Pascal plus Data Structures, Algorithms and Advanced Programming.* 2 ed., Lexington Ma: D.C. Heath, 1988.

Electronic Countermeasures, edited by Boyd, J.A., et.al. Los Altos: Peninsula, 1978.

Engelbrecht, G.N., *Numeric Simulation Techniques*, Simon's Town: SA Navy, 1996.

Erdogdu, F., *Arithmetic Architecture for Surface/Subsurface Bearing - Only Radar Tracking by Microcomputer.* M.Sc. research thesis , Naval Postgraduate School, Monterey, California. 1979.

Farina, A. and Struder, F.A., *Radar Data Processing - Volume 1 - Introduction and Tracking*. Letchworth: Research Studies Press, 1985.

Gunter, W.H., *Naval Infrared Handbook - Volume 1*. Simon's Town: IMT. 1996.

Harrison. D.A., *Radar Based Weapons System Simulation Model*. Simon's Town : IMT. Nov 1980.

Heikal, M., *The Road to Ramadan*, London: William Collins and Sons, 1975.

Herzog, C., *The War of Atonement*, Jerusalem: Steimatzky's Agency, 1975.

Hoover, S.V. and Perry, R.F., *Simulation - A Problem-Solving Approach*. Reading Ma: Addison-Wesley, 1989.

Insight Team of the Sunday Times, *Insight on the Middle East War*, London: André Deutch, 1974.

Insight Team of the Sunday Times, *The Falklands War*, London: André Deutch, 1982.

Instruments, Magnetism and Compasses, *Senior Commercial Pilot Licence Study Notes*, Half Way House: Avex Air Training (Pty) Ltd, 1988.

Jane's Fighting Ships 1967/68, edited by Capt J.E. Moore RN. London: Jane's Yearbooks, [1967].

Jane's Fighting Ships 1974/75, edited by Capt J.E. Moore RN. London: Jane's Yearbooks, [1974].

Jane's Fighting Ships 1995/96, edited by Capt R. Sharpe RN , London: Jane's Yearbooks, [1995]

Jane's Naval Weapon Systems, edited by E.R. Hooten, London: Jane's Yearbooks , 1996.

Jane's Weapon Systems 1980/81 edited by R.T. Pretty. London: Jane's Yearbooks,[1980].

Jordan, G.M., *Class Notes for a Short Course in Homing Guided Missiles*, Ann Arbor, Mi: FAAC. c1975.

Kennedy, W.J. and Gentle, J.E., *Statistical Computing*. New York: Marcel Dekker, 1980.

King, R.W.P. and Wu, T.T., *The Scattering and Defraction of Waves*. Cambridge: Harvard University Press, 1959.

Koburger, C.W., *Narrow Seas, Small Navies and Fat Merchantmen,* New York: Praeger, 1990
Lane, C.D. and Sleightholme, J.D., *The New Boatman's Manual.* Adlard Coles Ltd. 1967.

Law, A.M. and Kelton W.D., *Simulation Modeling and Analysis(sic.).* 2 ed. New York: McGraw-Hill, 1991.

Maloney, E.S., *Dutton's Navigation and Piloting,* 13 ed, Annapolis: Naval Institute Press, 1978.

Naval Operations Analysis. 2 ed. Annapolis, Ma: Naval Institute Press. 1977.

Perret, B., *Weapons of the Falklands Conflict,* Dorset: Blandford Press, 1982.

Phillips, L.D., *"Requisite Decision Modelling"*, Journal of the Operations Research Society, Vol 33, 1982.

Powers, M.J., Cheney, P.H. and Crow, G., *Structured Systems Development, Analysis, Design, Implementation.* 2 ed., Boston: Boyd & Fraser, 1990.

Richardson, D., *Naval Armament.* London: Janes. 1981.

Rossouw, J., *Review of Existing Wave Data, Wave Climate and Design Waves for South African and South West African (Namibian) Coastal Waters,* Stellenbosch : CSIR, 1984.

Skolnik, M.I., *Introduction to Radar Systems.* 2 ed., Auckland: McGraw-Hill, 1981.

SYSTAT for Windows: Statistics, Version 5 Edition. Evanston, IL: SYSTAT Inc, 1992.

Tabak, R., *Optimal Evasion of an Aircraft from a Coasting Proportional Navigation Guided Missile.* M.Sc. research thesis , Technion - Israel Institute of Technology. 1991.

Turbo Pascal Version 6.0 - Library Reference, Scotts Valley Ca: Borland, 1990.

Turbo Pascal Version 6.0 - Programmer's Guide, Scotts Valley Ca: Borland, 1990.

Van Brunt, L. B., *Applied ECM - Vol 1.* Dunn Loring, Va.: EW Engineering, c1978.

Van Brunt, L. B., *Applied ECM - Vol 2.* Dunn Loring, Va.: EW Engineering, c1982.

Von Clausewitz, Carl, *On War*. Edited with an introduction and notes by Anatol Rapoport. Harmondsworth, England : Penguin Books. [1832] 1968.

Watson, B.W., George, B., Tsouras, P., Cyr, B.C. and the International Analysis Group, *Military Lessons from the Gulf War*, London: Greenhill Books, 1991.

White Paper on Defence, Republic of South Africa, 1996.

Winston, W.L., *Operations Research - Applications and Algorithms.* 3 ed., Belmont Ca: Duxberry, 1994.

Zarchan, P., *Tactical and Strategic Missile Guidance*, 2 ed., Washington: AIAA, c1994.

Zucrow, M.J. and Hoffman, J.D., *Gas Dynamics Volume 1.* New York: John Wiley, 1976.