FOI
TOTALFÖRSVARETS
FORSKNINGSINSTITUT

FOI-R--0049--SE
Mars 2001
ISSN 1650-1942
Teknisk rapport

Mathias Brage

Störsimulering mot SAR

Utgivare	Rapportnummer, ISRN	Klassificering
Totalförsvarets Forskningsinstitut - FOI Avdelningen för Ledningssystemteknik Box 1165 581 11 Linköping	FOI-R--0049--SE	Teknisk rapport
	Forskningsområde 6. Telekrig	
	Månad, år Mars 2001	**Projektnummer** E7013
	Verksamhetsgren 5. Uppdragsfinansierad verksamhet	
	Delområde 61 Telekrigföring med EM-vapen och skydd	
Författare/redaktör Mathias Brage	**Projektledare** Veine Jernemalm	
	Godkänd av Leif Månsson	
	Tekniskt och/eller vetenskapligt ansvarig Veine Jernemalm	

Rapportens titel

Störsimulering mot SAR

Sammanfattning (högst 200 ord)

Att simulera olika system anses av såväl försvaret som industrin som en väl etablerad metod att spara både tid och pengar. Dagens snabba datorer gör det möjligt att simulera sådana experiment och situationer som tidigare behövde utföras med verklig mätutrustning. Genom att i förväg utföra en simulering kan värdefull kunskap erhållas för en viss situation och med hjälp av denna kunskap kan sedan det verkliga experimentet genomföras med större precision.

Vid institutionen för Telekrigssystem på Totalförsvarets Forskningsinstitut, FOI, i Linköping bedrivs forskning och utveckling kring olika tekniker att på elektronisk väg störa ut ett så kallat SAR-system. SAR står för *Synthetic Aperture Radar* och betecknar ett radarsystem vilket med hjälp av en konstgjord radarantenn skapar fotoliknande bilder av det markklotter som för andra radarsystem oftast är till problem.

Eftersom ett SAR-system kan operera oberoende av ljus- och väderförhållanden skulle ett sådant system i en krigssituation vara ett indirekt hot mot våra flygbaser, hamnar etc. Ett skydd mot detta hot vore att genom strategisk placering av en eller flera störsändare lägga ut en brusmatta i syfte att försvåra inhämtningen av viktig information.

Som ett led i utvecklingsarbetet av dessa SAR-störsändare behövs ett program med vilket relevanta störsituationer kan simuleras. Rapporten beskriver det examensarbete vars uppgift var att utveckla ett sådant simuleringsprogram.

Nyckelord

apertur, antennförstärkning, SAR, störsändare, störövervikt

Övriga bibliografiska uppgifter	Språk Svenska
Även utgiven som examensarbetesrapport vid Linköpings Universitet med reg nr: LiTH-ISY-EX-3146.	
ISSN 1650-1942	**Antal sidor:** 76 s.
Distribution enligt missiv	**Pris: Enligt prislista** **Sekretess:** Öppen

Issuing organization	Report number, ISRN	Report type
FOI – Swedish Defence Research Agency Division of Command and Control Warfare Technology Box 1165 SE-581 11 Linköping	FOI-R--0049--SE	Technical report
	Research area code 6. Electronic Warfare (EW)	
	Month year March 2001	**Project no.** E7013
	Customers code 5. Commission funded project	
	Sub area code 61 EW with EM-weapons & protection	
Author/s (editor/s) Mathias Brage	**Project manager** Veine Jernemalm	
	Approved by Leif Månsson	
	Scientifically and technically responsible Veine Jernemalm	

Report title (In translation)

Simulation for SAR jamming

Abstract (not more than 200 words)

The Defence Forces as well as the industry consider simulating different systems as a well-established method of saving both time and money. Fast computers of today make it possible to simulate those experiments and situations that previously needed to be performed with real equipment. The live experiment can be performed with higher precision when valuable knowledge is obtained by simulating a certain situation in advance.

The institution of Electronical Warfare at the Swedish Defence Research Agency is developing technologies for jamming SAR-systems. SAR stands for *Synthetic Aperture Radar* and describes a radar system that, with an artificial radar antenna, generates photo like pictures of the ground clutter that usually is a problem for ordinary radar systems.

Since a SAR-system can operate independently of light- and weather conditions such a system would in a war situation be an indirect threat for our air bases, harbours etc. Protection against SAR systems is obtained by strategic positioning of a set of jammers that will generate and transmit noise in order to prevent information collection.

As a part of the SAR-jammer development there is a need for a program in which relevant jam situations can be simulated. A description is given in this report of the master thesis work whose aim was to develope such a simulation program.

Keywords

aperture, antenna gain, SAR, jammer, SNR

Further bibliographic information	Language Swedish

Also issued as a master thesis report at Linköping University with reg no: LiTH-ISY-EX-3146.

ISSN 1650-1942	**Pages** 76 p.
	Price acc. to pricelist
	Security classification: Open

Förord

Denna rapport beskriver ett 20 poängs examensarbete utfört vid institutionen för Telekrigssystem på Totalförsvarets Forskningsinstitut, FOI, i Linköping. Detta arbete är slutprojektet på civilingenjörsutbildningen Datateknik med inriktning mot datorseende och grafik vid Linköpings Tekniska Högskola. Resultatet av examensarbetet är en prototyp för simulering och visualisering av störning mot SAR-system.

Författaren vid ett fältförsök med den "riktiga" störsändaren.

Författaren vill ta tillfället i akt och rikta ett stort tack till nedanstående personer utan vars hjälp detta arbete inte hade kunnat genomföras:

Veine Jernemalm (Handledare FOI), Göran Forsling (Adjunkt MAI), Stefan Hellman (Opponent), Erik Geijer-Lundin, Peter Knuts, Carl Tengstedt och Tobias Thörn (”Bollplank”), Ulf Henriksson (Examinator ISY).

Linköping, mars 2001

Mathias Brage

Innehåll

Figurförteckning

1 Inledning

I det här kapitlet förklaras det övergripande målet med projektarbetet samt varför en studie av ämnesområdet är av intresse. Ur en ganska vagt framställd målsättning definieras sedan i kapitel 3 en precis problemformulering, en problemformulering som sedan ligger till grund för det fortsatta arbetet.

Vidare kan detta inledningskapitel ses som en guide över innehållet i rapporten. I slutet av kapitlet ges därför en kortfattad beskrivning av upplägget på såväl arbetet som rapporten.

1.1 Bakgrund

Totalförsvarets Forskningsinstitut, FOI (tidigare FOA och FFA), som är en statlig myndighet under försvarsdepartementet har ca 1000 medarbetare fördelade på orterna Linköping, Stockholm och Umeå. FOIs verksamhet är inriktad på forskning och utredningsarbete för totalförsvaret samt till stöd för nedrustning och internationell säkerhet. Som uppdragstagande myndighet åtar sig FOI specificerade forskningsuppdrag från kunder såsom försvarsmakten, försvarets materielverk, försvarsdepartementet, utrikesdepartementet, överstyrelsen för civil beredskap samt statens räddningsverk. Forskningen sker i projektform på någon av de åtta avdelningarna Människa-System-Interaktion, Vapen och Skydd, NBC-skydd, Försvarsanalys, Sensorteknik, Ledningssystemteknik, Flygteknik eller Systemteknik.

Avdelningen för Ledningssystemteknik är placerad i Linköping och bedriver i huvudsak forskning inom områdena ledningssystem och ledningskrigföring. Inom området för ledningssystem är tekniker för effektiv och säker informationsöverföring, informationsbehandling samt datafusion av särskilt stort intresse eftersom dagens ”IT-fierade” försvar får en allt starkare hotbild även inom dessa moment. För området ledningskrigföring är forskningen till största delen fokuserad kring signalspanings- och motmedelssystem för radio och radar. Vidare gäller att eftersom simulering av olika system är en så pass viktig komponent inom forskningen, ingår därför även forskning på metoder och tekniker för simulering som en stor del av den totala forskningsverksamheten inom FOI.

Under avdelningen för Ledningssystemteknik ligger institutionen för Telekrigssystem, där forskning sker kring den teknik som brukas eller kommer att brukas inom telekrigföring. Telekrigföring benämner de åtgärder och metoder som utnyttjas för att upptäcka, utnyttja, påverka, försvåra eller förhindra en motståndares användning av telemedel. Samtidigt gäller det att på ett tillräckligt säkert och effektivt sätt kunna upprätthålla den egna användningen av telemedel. De telemedel som idag är av störst intresse är de medel som används i de elektromagnetiska spektrumen som exempelvis IR, laser, radio, radar samt UV.

Telekrigföring är således de åtgärder som riktas mot den sensorteknik och/eller de sensorsystem som används i avseende att kommunicera, navigera, varna, spana, styra osv. En allt mer ökande användning av signalspanings- och motmedelssystem i de militära tillämpningarna har satt allt större krav på de telesystem som används vad gäller skydd och säkerhet. Även de civila, och då främst de kommersiella, användarna av teletekniska hjälpmedel efterfrågar allt mer störsäkra och avlyssningsskyddade system.

Mot bakgrund av detta bedriver därför institutionen för telekrigssystem ett flertal forskningsprojekt inom områdena radar-/radiospaning samt radar-/radiostörning. Det projekt som detta examensarbete lyder under syftar till att störa ut ett så kallat SAR-system (*Synthetic Aperture Radar*). Den ursprungliga uppgiftsformuleringen för examensarbetet lyder:

”I en krigssituation skulle fientliga SAR-system kunna vara ett indirekt hot mot våra flygbaser, hamnar, militära grupperingar mm. Genom att t ex utnyttja störsändare kan man försvåra för en fiende att inhämta information. En möjlig metod är, att med en eller flera störsändare lägga ut en brusmatta för att maskera intressanta markmål.

Uppgiften består i att utifrån givna data på störsändare, radarsystem, geometriska förhållanden mm ta fram ett datorprogram med vars hjälp relevanta störsituationer kan simuleras.”

1.2 Omfattning

Rapporten beskriver i stort sett utvecklingsarbetet i kronologisk ordning med förklaringar av de utförda moment som lett fram till ett, tills vidare tillräckligt, fungerande simulatorsystem. Vissa undantag har dock gjorts för att göra framställningen mer logisk. Vidare ges kortfattade beskrivningar av metoder, delsystem, förenklingar etc där dessa anses nödvändiga för att underlätta läsförståelsen.

I rapporten görs däremot inga som helst anspråk på att i detalj ge fullständiga härledningar av matematiska formler eller liknande. Där formler förekommer kan viss härledning förekomma annars ges en hänvisning till mer detaljerad information.

1.3 Syfte

För att kunna avlägga en civilingenjörsexamen vid LiTH fordras att den studerande har utfört ett godkänt examensarbete. Examensarbetet för en civilingenjörsutbildning omfattar 20 poäng på D-nivå, vilket motsvarar en termins arbete på ca 20 effektiva arbetsveckor.

1.3.1 Syfte med examensarbete

Syftet med ett examensarbete i allmänhet är att den studerande skall kunna uppvisa att denne:

- Kan använda sina under studietiden förvärvade kunskaper vid lösning av en förelagd uppgift med anknytning till utbildningen.
- Har god förmåga till professionell skriftlig och muntlig kommunikation.
- Kan tillgodogöra sig innehållet i relevant facklitteratur på området och sätta sig in sitt arbete i detta sammanhang.
- Kan kritiskt granska och diskutera ett i tal och skrift framlagt examensarbete.
- Kan använda kunskaper och färdigheter som erhållits vid studier på fördjupningsnivå C (60-poängsnivå) och D (80-poängsnivå).

Syftet med just detta examensarbete i synnerhet kan formuleras enligt följande:

Uppgiften består i att bygga ett datoriserat simulatorsystem som på lämpligt sätt beskriver förhållandet mellan ett SAR-system och ett antal störsändare.

1.3.2 Syftet med rapporten

Syftet med denna rapport är att dokumentera examensarbetet genom att:

- Förklara och precist definiera den problemformulering som utgjort grundstommen för själva arbetet.
- Motivera val av lösningsmetodik samt förklara tillvägagångssätt av utvecklingsarbetet.

– Sammanställa de åtgärder som vidtagits för att uppnå de dellösningar som krävts för att lösa uppgiften given i problemformuleringen.

– Presentera resultat, slutsatser samt framtida förbättringar.

1.4 Målgrupp

Denna rapport riktar sig i första hand till personer med civilingenjörsutbildning eller motsvarande, och då främst med inriktning mot datateknik och elektronik. Kunskaper inom antennteori, vågutbredningslära, modellering och programmering underlättar för att till fullo kunna tillgodogöra sig innehållet i rapporten.

1.5 Upplägg

Nedan följer en beskrivning av dels själva arbetets upplägg och dels av rapportens upplägg. Avsnittet om rapportens upplägg kan ses som en läsanvisning för den som önskar en kort översikt av vad ett enskilt kapitel behandlar.

1.5.1 Arbetets upplägg

Arbetet inleddes med att göra en ingående studie av de SAR-system som idag finns i bruk och vilka är beskrivna i litteraturen. Vidare har de metoder och den teknik som är vanligast förekommande i störsammanhang studerats. Detta för att få en uppfattning om hur förhållandet mellan radar och störsändare ser ut i praktiken, vilket är av avgörande betydelse för att kunna konstruera en simulator med så hög trovärdighet som möjligt.

Den större delen av arbetstiden har dock varit av problemlösningskaraktär. Här har matematiska, modellerings- och programmeringsproblem i huvudsak stått i centrum. Viktigt att poängtera är att arbetet inte har fortskridit så tillvida att ett moment har strikt avhandlats före ett annat. En bättre beskrivning vore att se det hela som en iterativ process där ovanstående tre moment har mixats om vartannat för att med regelbundna tester kontrolleras så att ett önskat delresultat har uppnåtts.

Slutligen har resultatet av arbetet sammanställts i denna rapport. Nämnas kan att grovt räknat har ca 10% av tiden ägnats åt litteraturstudier, 60% åt problemlösning och implementering samt resterande 30% har ägnats åt rapportskrivande.

1.5.2 Rapportens upplägg

I rapporten förekommer en del förkortningar och för området specifik terminologi. För att underlätta för läsaren har ett index med de vanligast förekommande förkortningarna och termerna sammanställts i slutet av rapporten. I övrigt ges den översiktliga beskrivningen av varje enskilt kapitel enligt nedan.

Kapitel 2: Förutsättningar. Här ges en beskrivning av de principer och förutsättningar som ligger till grund för de delsystem som skall simuleras. Vidare kommer en viss förklaring att ges av den del av begreppsapparaten som fortsättningsvis används i rapporten.

Kapitel 3: Problemformulering. Den vaga uppgiftsformuleringen analyseras och en precis problemformulering definieras och presenteras. Simuleringsprogrammets inparametrar beskrivs och i slutet av kapitlet ges en redogörelse av de förenklingar som gjorts.

Kapitel 4: Implementering. Kapitlet ger en översikt av programmets moduluppdelning samt beskriver kopplingen mellan moduler och de problem specificerade i kapitel 3.

Kapitel 5: Resultat. Här presenteras resultaten för några illustrativa störsituationer.

Kapitel 6: Avslutning. En avrundning av examensarbetet görs genom att presentera slutsatser samt kända fel. Dessutom ges avslutningsvis några förslag på framtida förbättringar.

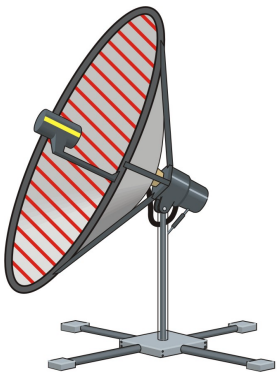
2 Förutsättningar

Utvecklingen inom signal- och informationsbehandling har inneburit att programmerbara och flexibla radarsystemfunktioner kunnat införas till rimliga kostnader. Man har bl a inriktat sig på att undertrycka störande signaler från bakgrundsmiljön och från olika radarmotmedel samt att utnyttja olika målsignaturer för klassificering. Här kan nämnas att doppler- och MTI-metoder (*Moving Target Indication*) har kunnat effektiviserats för att bortsortera störekon. Vidare har konstruktion av radar med hög upplösning i avstånd och i sida blivit möjlig med hjälp av pulskompression- respektive SAR-teknik. Detta medför i sin tur ökad möjlighet att på ett säkert sätt kunna fastställa samt klassificera olika måltyper.

Tack vare tekniska framsteg av detta slag har radarspaning mot mål på mycket stora avstånd blivit mer intensifierad. Höghöjds- och satellitspaning tillämpas i allt större utsträckning inom såväl civil som militär verksamhet sedan tillgången på snabba, billiga datorer ökat.

2.1 SAR-egenskaper

I radarteorin gäller att form och storlek på loben vid målet bestäms av utformning och storlek på radarantennens så kallade apertur. Apertur är benämningen på den ”öppning”, alternativt tvärsnittsyta, i radarantennens reflektor där radarpulserna strålar ut och tas emot, se Figur 2.1. Här råder ett omvänt proportionalitetsförhållande mellan lobstorlek och aperturstorlek. En antenn med en stor apertur ger en smalare lob i jämförelse med en antenn vars apertur är mindre. Lobstorleken är av avgörande betydelse för radarns precision i höjd och i sida, ty en smal lob resulterar i en högre upplösning och därmed bättre särskiljningsförmåga jämfört med en bredare lob, se vidare Kingsley & Quegan (1992) och Levanon (1988).

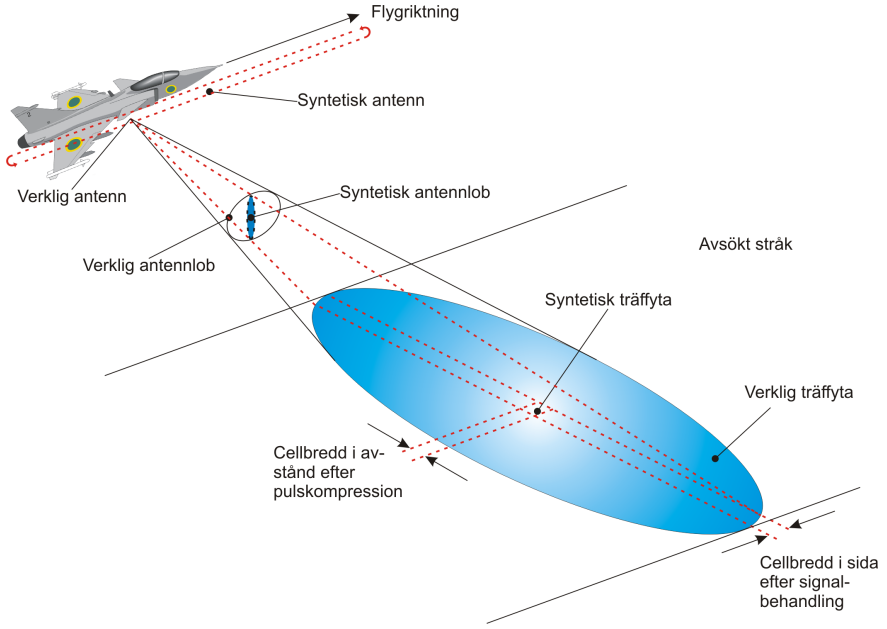


Figur 2.1 *Antennens apertur är rödstreckad.*

Grovt kan sägas att ju smalare loben är desto högre blir upplösningen, men även radarantennens användningsområde bestämmer utformningen

på aperturen och således loben. Som exempel kan nämnas att i navigeringssammanhang till sjöss används en bred cigarrformad radarantenn, s k navigationsradar, där upplösningen i sida (horisontalplanet) är viktigare än upplösningen i höjd (vertikalplanet).

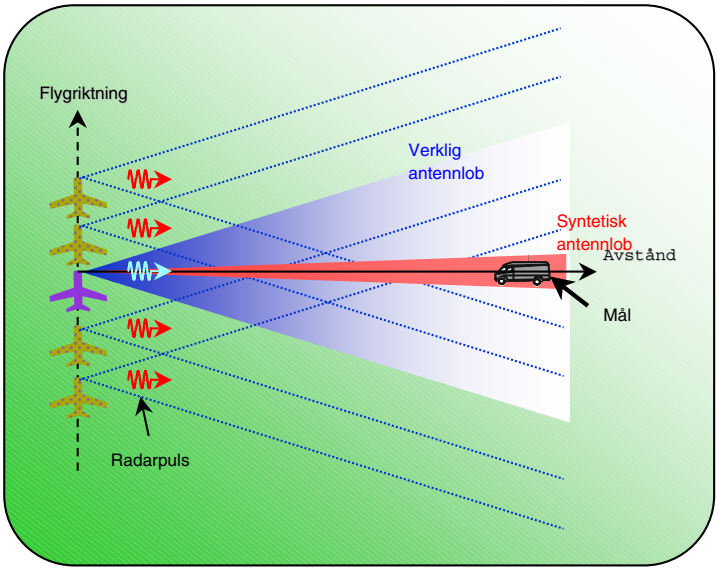
Som nämndes inledningsvis så står förkortningen SAR för *Synthetic Aperture Radar*. SAR är således benämningen på ett radarsystem som med hjälp av avancerad signalbehandling kan syntetisera en extremt lång virtuell apertur i syfte att förbättra upplösningen i den önskade målbilden, se Figur 2.2. Brist på utrymme ombord på flygplan eller satelliter gör det problematiskt att använda sig av stora antenner, och därför använder man sig idag av SAR-tekniken för att med en mindre antenn uppnå önskad upplösning. Det första fungerande SAR-systemet utvecklades och byggdes dock redan på 50-talet av ett forskarlag lett av den amerikanske vetenskapsmannen Carl Wiley. Wileys upptäckt har lett till att SAR-tekniken idag är den ledande radartekniken som används för att övervaka jordens resurser. Dagens system är i stort sett baserad på samma princip som Carl Wileys första fungerande SAR-system.



Figur 2.2 *SAR-systemets grundprincip.*

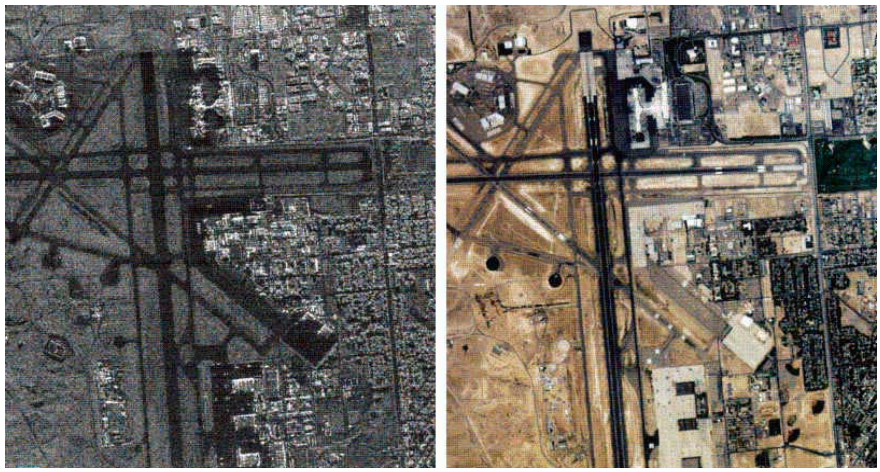
Grundprincipen för SAR är att låta en flygburen radar förflytta sig rätlinjigt längs en s k integrationssträcka och insamla en uppsättning ekosignaler. Dessa signaler digitaliseras, faskorrigeras och lagras för att senare medelvärdebildas. En SAR-bild beräknas först då en integrationssträcka har tillryggalagts och en medelvärdesbildning kan utföras. Resul-

tatet efter medelvärdesbildningen blir en signal som har karaktären av en signal genererad av den mycket större antennen, dvs en virtuellt smalare radarlob har erhållits och därmed högre upplösning i sida, se Figur 2.3. På detta sätt kan en liten antenn få en konstgjord lob som har lika bra upplösning som en lob från en stor antenn. Synthetic Aperture Radar är således en radar där storleken på radarantennens apertur i verkligheten är relativt liten, men som genom ovanstående metod blir tillräckligt stor för att åstadkomma önskad storlek på radarloben. Det finns SAR-system som kan urskilja objekt med en bredd ner till 30 cm.



Figur 2.3 *SAR-systemet ombord på flygplanet skickar iväg ett antal radarpulser medan det färdas längs en rät linje. Dessa pulser läggs ihop till en större puls som ger en mycket smalare antennlob.*

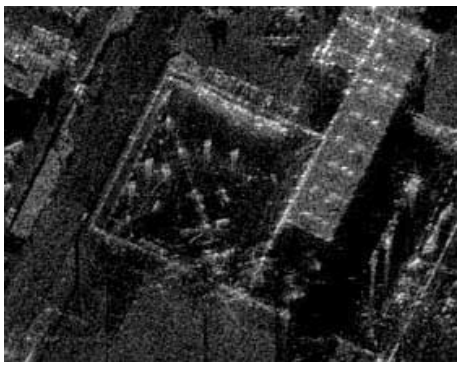
Medan SAR används företrädesvis i flygburna farkoster för spaning mot mer eller mindre fasta mål används ISAR, Inverse SAR, för spaning mot rörliga mål, t ex fartyg och flygplan. Eftersom SAR och ISAR är okänsliga för oönskade bakgrundssignaler kan fotoliknande bilder åstadkommas av en ytas topografi oberoende av väder- eller ljusförhållande. Dessa SAR-bilder är numera i stort sett nästan lika detaljerade som vanliga kamerabilder, se Figur 2.4.



Figur 2.4 *Flygplats avbildad med SAR-teknik till vänster och optiskt foto till höger. SAR-bilden som är tagen på ca 2-15 km avstånd har en ungefärlig upplösning i sida på 3 m.*

Detaljnivån i en SAR-bild beror dels på upplösningen i sida och dels på upplösningen i avstånd. Upplösningen i sida är, enligt tidigare resonemang, begränsad av radarantennens lobstorlek och för SAR gäller generellt att en mindre antenn ger en smalare lob, dvs det omvända mot vad som gäller för en vanlig radarantenn. Vidare erhålls bra upplösning i avstånd genom stor bandbredd och högt pulskompressionsförhållande.

2.2 Störsändaregenskaper

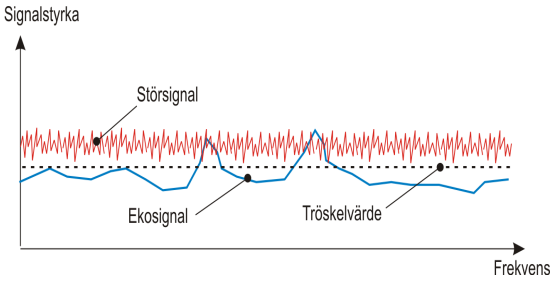


Figur 2.5 *Ett flygplan i sin hangar kan upptäckas med SAR-teknik.*

För vanliga radarsystem gäller det att hitta mål i en klottrig miljö, dvs där det finns oönskade bakgrundsreflexer, medan motsatsen gäller för SAR; det är just klottret som är informationen. SAR-system som använder sig av lägre frekvenser i exempelvis VHF-området kan lättare penetrera vegetation och maskeringar tack vare den större våglängden. Följden av detta blir att effektiv maskering av intressanta objekt blir mycket svår om traditionell kamouflering används, se Figur 2.5. Genom strategiskt placering av en eller flera störsändare i närheten av det känsliga området kan dock fiendens informationsinhämtning försvåras.

Beroende på vilken störverkan som skall åstadkommas, maskerande eller vilseledande, ges störsändaren olika egenskaper. I det fall man avser att förhindra upptäckt används en störsändare av maskerande typ, vilken avser att dölja de verkliga radarekona med hjälp av brus.

Eftersom alla elektroniska komponenter genererar brus av varierande styrka kan detta utnyttjas för att dölja ett radareko. En radar som utsätts för störning med vitt Gaussiskt brus har ingen möjlighet att kunna skilja detta från det egengenererade bruset. Om det inkommande bruset är tillräckligt starkt kan de verkliga radarekona inte filtreras ut och följaktligen ej heller detekteras. Ekona sägs ”drunkna” i bakgrundsbruset, se Figur 2.6. Då vitt Gaussiskt brus och fasbrus är relativt enkelt att generera i exempelvis en mikrodator, är dessa två typer vanligast förekommande vid störning med brus. Fasbrus genereras t ex genom att med hjälp av en slumpad binärkod fasvrida den inkommande radarpulsen 180° för, säg, varje 0:a.

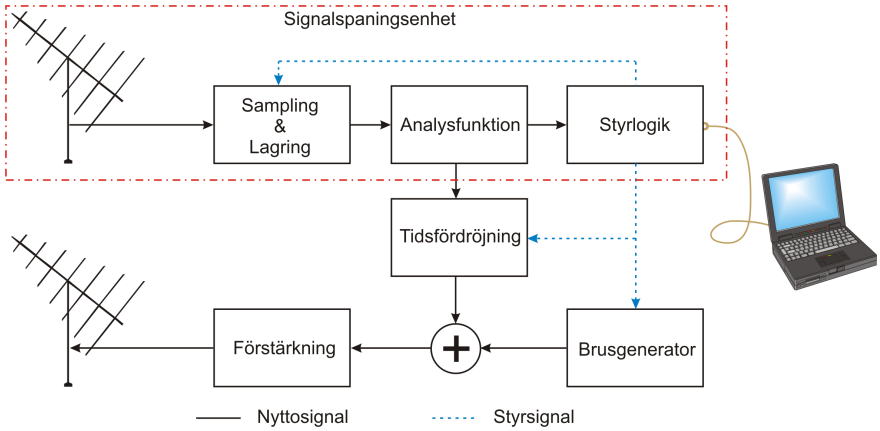


Figur 2.6 *Ekosignalen drunknar i bakgrundsbruset då styrkan på detta blir för stort.*

För att kunna uppnå största möjliga störverkan är det viktigt att tänka på att störningen sker på rätt frekvens, i rätt tid och i rätt riktning. Störverkan kan åstadkommas genom såväl passiva som aktiva störmetoder. En passiv störmetod genererar inte någon egen energi utan reflekterar eller absorberar enbart energin utsänd från radarn. Ett exempel på en vanligt förekommande passiv störmetod är så kallad ”remsfällning” där en stor mängd metallöverdragna remsor sprids ut i form av ett moln. Detta remsmoln ger upphov till skenekon som kan maskera ett verkligt mål och därmed vilseleda en radar. Vidare är även en metod som går under benämningen ”smygteknik” en passiv störmetod där en farkosts utformning och konstruktionsmaterial anpassas i syfte att minska radarmålytan och därmed uppnå en kamouflerande effekt.

En aktiv störmetod däremot kan exempelvis vara att kontinuerligt sända ut störsignaler över ett visst frekvensspektrum. Nackdelen med en sådan metod är att om störobjektet oregelbundet byter frekvens, s k frekvenshopp, måste störsändaren upprätthålla en konstant störning över ett relativt brett frekvensband. Detta medför en stor energiåtgång samt att även andra mottagare inom samma frekvensområde kan bli utstörda. En lösning på det problemet är att använda sig av en så kallad repeterstör-

sändare. En störsändare av denna typ kan vara en anordning bestående av tre betydande delar; en mottagardel, en sändardel och en mellanliggande signalbehandlings- och kontrolldel (mikrodator el. motsvarande), se Figur 2.7. Mottagardelen, som kan liknas vid en signalspaningsenhet, läser av en radarpuls med avseende på pulsens frekvens, varaktighet, repetitionsfrekvens samt effekt. Radarpulsen spelas in, brusmoduleras och eventuellt tidsfördröjs innan den återutsänds.



Figur 2.7 *Principskiss för en maskerande brusstörsändare av repetertyp.*

Den utsända störpulsen kan göras olika lång beroende på hur stort område som skall maskeras. För att åstadkomma detta kan två eller flera av den inspelade radarpulsen repeteras omedelbart efter varandra varefter brus påmoduleras.

En annan viktig fördel med att utnyttja repeterande störsändare är att störningen bara sker under den tid då en fientlig radarspaning är aktiv. Genom att på detta vis begränsa den egna sändningen, minskas möjligheten för fienden att pejla in och bekämpa störsändarna.

3 Problemformulering

Uppgiften för examensarbetet är att utifrån scenariot beskrivet i 2.2 konstruera ett datorprogram med vars hjälp relevanta störsituationer kan simuleras. Inparametrar till programmet skall vara radarns och störsändarnas egenskaper, lägeskoordinater etc. Vidare är information om markdämpning av intresse. Programmet skall utifrån givna indata beräkna den aktuella störövervikten för olika delar av det avsökta området.

Med störövervikt avses här storheten:

$$\frac{J}{S} = \frac{\textit{mottagen effekt från störsändaren}}{\textit{mottagen effekt från målet}}$$

3.1 Tolkning av störövervikten

Om uttrycket för störövervikt ses utifrån ett detekteringsperspektiv så skall kvoten mellan effekterna tolkas som ett vilkor för att detektera antingen ekoeffekt eller störsändareffekt. Då värdet på störövervikten överstiger 1 så är den mottagna störsändareffekten, *J*, större än den mottagna ekoeffekten, *S*, och radarsystemet detekterar störsignalen istället för ekosignalen. Avsedd störverkan har alltså uppnåtts.

I Hyberg (1993) ges förslag på uttryck för de båda effektstorheterna *J* och *S* vilka skall utgöra grundstommen vid framtagningen av en lämplig formel att använda vid beräkning av störövervikten. Dessa är:

$$J = \frac{P_j \cdot G_j}{4\pi \cdot R_j^2} \cdot \frac{1}{\alpha} \cdot \frac{G_r \cdot \lambda^2}{4\pi} \quad \text{och} \quad S = \frac{P_r \cdot G_r}{(4\pi \cdot R_r^2)^2} \cdot \sigma \cdot \frac{G_r \cdot \lambda^2}{4\pi},$$

där *j* = jammer (störsändare), *r* = radar och de ingående komponenterna har följande betydelse:

G_j = Störsändarens antennförstärkning i den riktning mot mottagaren där störövervikten ska beräknas, dimensionslös.

G_r = Radarsystemets antennförstärkning i den riktning mot störsändaren där störövervikten ska beräknas, dimensionslös.

$\frac{G_r \cdot \lambda^2}{4\pi}$ = Radarantennens effektiva area, ofta 10-60 % mindre än den fysiska antennarean. λ = radarpulsens våglängd.

P_j = Den uteffekt med vilken störsändaren sänder.

P_r = Den uteffekt med vilken radarsystemet sänder.

$4\pi \cdot R_j^2$ = Avståndet mellan radar och störsändare bestämmer storleken på den sfäriska area där störsändarens uteffekt fördelas.

$(4\pi \cdot R_r^2)^2$ = Avståndet mellan radar och mål bestämmer storleken på den sfäriska area där radarsystemets uteffekt fördelas. Den yttre kvadraten förklaras av att pulsen färdas sträckan mellan radar och mål tur och retur.

$\frac{1}{\alpha}$ = Dämpnings- eller förstärkningsfaktor. Radarsystemets antennförstärkning, G_r, förändras för varje sidolob med en för sidoloben tillhörande dämpnings-/förstärkningsfaktor.

σ = Föremålets effektivarea, den så kallade radarmålytan. Faktorn indikerar hur stort målet verkar vara när det betraktas i radarsystemet.

Beräkning av effektförhållandet sker enligt Hybergs (1993) uttryck för störövervikt vilket lyder:

$$\frac{J}{S} = \frac{4\pi \cdot R_r^4 \cdot P_j \cdot G_j \cdot \beta}{R_j^2 \cdot P_r \cdot G_r \cdot \alpha \cdot \sigma}.$$

Uttrycket är något omarbetat för att ge en bättre överblick, men de ingående komponenterna är de samma. Ytterligare en faktor, β, har dock tillkommit vilken illustrerar hur störsändarens uteffekt minskas i takt med förhållandet mellan ekopulsens och störpulsens respektive bandbredder. Bandbreddsförhållandet, β:s, påverkan förklaras mer ingående i 3.3.3.

3.2 Önskat resultat

Programmet skall beräkna och presentera styrkan av störsändarnas sammanlagda störövervikt längs den integrationssträcka som SAR-systemet har tillryggalagt. Presentationen av resultatet skall åskådliggöras grafiskt genom att styrkan hos störövervikten inom området representeras av olika färger. Ur presentationen skall vidare SAR-systemets rörelseriktning, störsändarnas placering samt störöverviktens utbredningsområde tydligt framgå.

3.3 Önskvärda inparametrar

I examensarbetets uppgiftsformulering finns ett förslag på ett antal önskvärda inparametrar. Dessa lyder enligt följande:

SAR: Flygbana, höjd, avstånd till belyst yta, integrationsfaktor, uteffekt, pulsrepetitionsfrekvens, pulskompressionsförhållande, antenndiagram.

Störsändare: Lägeskoordinater, uteffekt, antenndiagram, bandbreddsförhållande mellan radar och störsändare, procentuell störtid av radarpulsen.

Övrigt: Markdämpning samt hänsyn till sträckdämpning, dvs signalens dämpning som funktion av avståndet.

För programmets funktionalitet är vissa av de uppräknade inparametrarna särskilt betydelsefulla medan andra helt saknar betydelse. Efter noggrant funderande har betydelsen för några av inparametrarna omformulerats, ytterligare några viktiga inparametrar har tillförts listan samt de obetydliga inparametrarna har slopats. Den slutgiltiga parameterlistan med en kort parameterbeskrivning presenteras nedan.

3.3.1 SAR

Höjd: Den arbetshöjd som ett SAR-system kan tänkas befinna sig på då den aktuella informationsinhämtningen sker. Det av FOA utvecklade CARABAS-systemet har haft en arbetshöjd på ca 3000–4000 m i de fältförsök som institutionen för telekrig deltagit i. CARABAS = Coherent All Radio Band Sensing.

Avstånd till belyst yta: Det horisontella avståndet mellan farkostens lodlinje och aktuellt avsökningsområde. Ett typiskt värde för CARABAS är ca 2700 m.

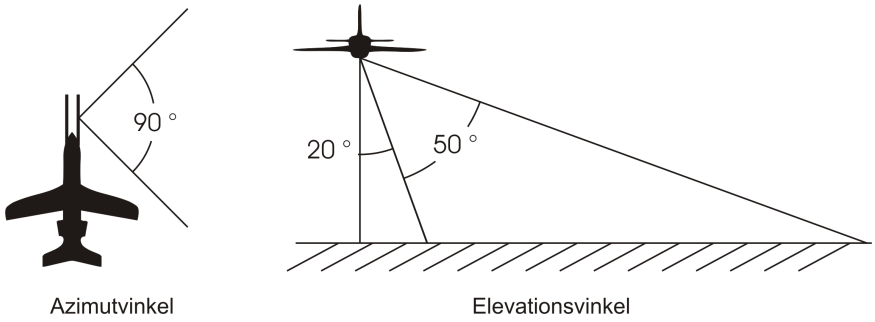
Uteffekt: Effekten med vilken radarpulserna sänds ut. CARABAS utnyttjar en ungefärlig uteffekt på ca 300 W.

Pulslängd i mikrosekunder: Längden på pulsen avgör hur bra upplösningen i avstånd ett radarsystem kan erhålla, ju kortare desto bättre. Detta står dock i konflikt med den effekt en puls sänds ut med; ett högt effektuttag under en kort tidsperiod kan generera strömöverslag vilket förstör komponenterna i utrustningen. För att förhindra detta används så kallad pulskompression, där pulsen byggs upp under en lite längre tidsperiod med hjälp av exempelvis frekvensmodulation, enligt Kingsley & Quegan (1992).

Antenndiagram: En exakt analytisk formel utan förenklingar existerar inte för en radars antennförstärkning i horisontal- och vertikalplanets alla tänkbara riktningar. Men för att ändå kunna efterlikna radarantennens uppförande så verklighetstroget som möjligt kan en antenns förstärkning i olika riktningar mätas upp och sammanställas i ett så kallat antenndia-

gram. Grunden till denna förenkling är att om en radarantenns lober delas upp i jämna vinkelintervall kan dessa ses som gränser inom vilka effekten antas vara konstant. Genom att i antenndiagrammet återspegla förstärkningen i en viss riktning erhålls en uppfattning om hur en antenns lobutformning ser ut i just den riktningen. Viktigt att tänka på är att ju fler värden som används i diagrammet desto mer verklighetsanpassat blir resultatet från simuleringen.

CARABAS-systemet använder sig av två horisontellt placerade, sprötliknande antenner, så kallade dipolantenner. Fördelen med användandet av två antenner med horisontell polarisation är att signaler från fel område kan undertryckas genom att beräkna tidsdifferensen. Nackdelen är att en dipolantenn ger stora lobvinklar i såväl horisontal- som vertikalplanet. En dipolantennen har en rundstrålande karaktär tvärs antennen, dvs antennen har en elevationsvinkel på 360° i vertikalplanet. I CARABAS-systemet har dock denna elevationsvinkel begränsats till intervallet 20°-70° i förhållande till farkostens lodlinje. Dessutom utnyttjas en azimutvinkel på 90° i horisontalplanet, se Figur 3.1.



Figur 3.1 *Lobutformning i CARABAS-systemet.*

3.3.2 Störsändare

Position: När en störsändare placeras ut skall dess respektive position kunna anges i förhållande till något slags referenssystem.

Uteffekt: Effekten som störpulserna sänds ut med. I störförsöken har en uteffekt på ca 20–40 W använts, vilket visat sig vara mer än tillräckligt. I störsammanhang anses det vara lämpligt att låta störövervikten uppgå till ett par decibel för att erhålla en tillräckligt effektiv störverkan, dvs mottagen störeffekt är ungefär dubbelt så stor som den mottagna ekoeffekten. Med denna tumregel som riktlinje har därför ett värde på ca 7 W ansetts vara mer rimligt vid störning av CARABAS.

Antal repetitioner: Beroende på hur stort stördjup man vill åstadkomma kan två eller flera inspelade radarpulser ”skarvas” ihop till en längre

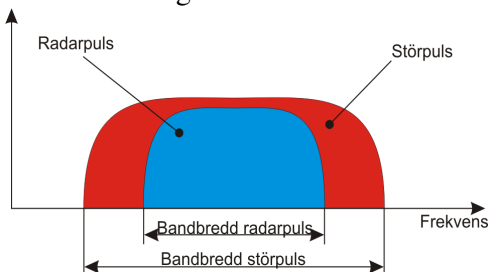
puls. Längre störtid medför längre maskerad sträcka enligt: $s = c \cdot t$, där c är ljushastigheten. Ett typisk värde för störsändare att använda sig av i CARABAS-samanhang är 1–5 pulser.

Procentuell störtid: Inom vissa frekvensband kan det på grund av så kallad ”rundgång” i systemet vara svårt att lyssna och störa samtidigt. För att garantera att störning av en radarpuls sker under rätt tidsintervall kan viss intelligens byggas in i störsystemet. Detta görs genom att dela upp lyssnings- och sändarperioderna i flera mindre delintervall för att på så vis erhålla ett lyssna-sänd-mönster. Antalet delintervall och storleken på dessa kan efter behov ändras, men ju mindre den procentuella sändningsdelen blir desto mindre blir också den totala uteffekten i enlighet med att $P = W/t$, där W är använd energi. Med anledning av detta används den procentuella störtiden som en dämpningsfaktor vilken ligger i intervallet [0, 1].

Antenndiagram: På samma sätt som för att kunna återspegla en radars lobkaraktäristika skall en störsändares antenndiagram kunna ges som inparameter till programmet.

3.3.3 Övrigt

Bandbreddsförhållande: Definitionsmässigt är bandbreddsförhållandet resultatet av radarpulsens bandbredd dividerat med störpulsens bandbredd och liksom procentuell störtid är denna storhet en slags dämpningsfaktor. För att med säkerhet eliminera den utsända radarpulsen bör störpulsens bandbredd helt överlappa radarpulsens bandbredd, dvs bandbreddsförhållandet är lika med 1. I verkligheten måste dock en viss marginal införas då det är mycket svårt att generera en störpuls som har exakt samma bandbredd som radarpulsen, se Figur 3.2. Det viktiga i sammanhanget är dock att bandbreddsförhållandet inte får bli för litet, ty detta bidrar i sin tur till att störövervikten minskar.



Figur 3.2 *Radarpulsens bandbredd överlappas av störpulsens bandbredd.*

Markdämpning: Beroende på det avspanade områdets absorptions- och spridningsegenskaper kommer energin från en utsänd radarpuls att dämpas olika. Som exempel kan nämnas att vatten totalt sett har en mindre dämpningsfaktor och därmed reflekterar mer energi än torr ökensand. I vågutbredningssammanhang förekommer även atmosfärisk dämpning

men enligt Stimson (1998) är denna dämpning mycket liten för frekvenser under 0.1 GHz. Eftersom CARABAS använder frekvenser i bandet 20-88 MHz, så har den atmosfäriska dämpningen i störförsöken försummats och markens dämpningsfaktor har uppskattats till ca 30 dB.

3.4 Förenklingar och beräkningar

Värt att notera är att ovan angivna värden på respektive inparameter har legat till grund för hur simulatorprogrammet har konstruerats. Tanken är naturligtvis att en simulering med dessa värden skall ge ett med verkligheten tillräckligt överensstämmande resultat. Men för att kunna konstruera en modell av ett verkligt händelseförlopp har en del förenklingar varit nödvändiga att införas. I ett senare skede är det dock önskvärt att modellen ska kunna byggas ut för att bli mer verklighetstrogen.

På grund av att beräkning och presentation av en uppsättning samverkande störsändares störövervikt är programmets huvuduppgift så har den valda avgränsningen lett till att funktionalitet av mindre signifikans givits mindre prioritet eller helt uteslutits. Nedan följer en sammanställning av de nödvändigaste förenklingarna och begränsningarna som införts i programmet.

3.4.1 Omgivning

- Eftersom det i modellen inte kommer existera några andra föremål än själva störsändarna så är en plan markyta det enda måleko som kan förekomma. Genom denna förenkling tillåts alla målekon med samma avstånd returnera samma effekt och beräkningen av störövervikten i varje punkt längs flygbanan blir särskilt enkel. Beräkningen i en viss punkt (en pixel på skärmen) görs genom att endast den största mottagna ekoeffekten jämförs med den mottagna störsändareffekten. Som framgår av uttrycket beskrivet i 3.1 har den mottagna ekoeffekten ett omvänt avståndsberoende vilket betyder att den största mottagna ekoeffekten återfinns vid det kortaste avståndet, dvs inga starkare ekon kan förekomma bortanför det kortaste avståndet då den plana ytan endast tillåts ha en homogen markdämpning. Eller annorlunda uttryckt; är störövervikten större än 1 i aktuell punkt så visas den mottagna störsändareffekten representerad med lämplig färg, annars vissas den plana ytan som representerar marken.

- I dagsläget finns det av prioriteringsskäl ingen funktion som stödjer utmärkandet av ett eventuellt skyddsområde. Däremot har det avsökta stråket utefter den tänkta flygbanan markerats för att presentationen av störövervikten skall bli mer begriplig.

3.4.2 Radar

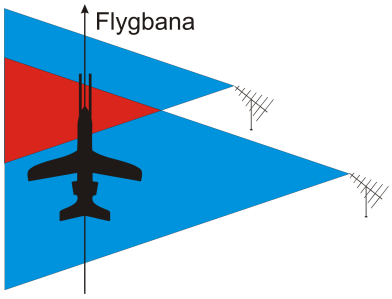
- Som beskrivet i 3.3.1 vore det önskvärt om ett radarsystems antenndiagram kunde ingå vid beräkningen av störövervikten. Men för att minska beräkningsarbetet används till en början en förenkling där detekteringsområdet för radarantennen utgörs av störsändarnas verkansområde. Enkelt uttryckt betyder detta att effekten av störsändningen börjar detekteras när radarsystemet träder in i en godtycklig störsändares verkansområde. En störsändares verkansområde blir det område som täcks in av störsändarens antenndiagram. Med denna förenkling kommer programmet kunna hantera två fall; det första fallet är en allmän huvudlob beskriven som en godtycklig kon, medan det andra fallet grundar sig på CARABAS karaktäristiska huvudlobsutformning återgiven i Figur 3.1. Att med en matematisk formel beskriva utseendet på en huvudlob är synnerligen praktiskt vid beräkning av den träffyta som uppstår då loben träffar marken.

- En annan förenkling på radarsidan är att radarsystemet förflyttar sig enbart längs den vertikala axeln i ett koordinatsystem. Denna förenkling visade sig vara det enklaste sättet att beskriva radarsystemets flygbana i ett enhetligt referenssystem. På detta vis förenklas modellerings- och beräkningsarbetet avsevärt men tyvärr på bekostnad av programmets flexibilitet. Möjligheten för en användare att godtyckligt kunna lägga in en lämplig flygbana kommer eventuellt att byggas in i en senare version av programmet.

3.4.3 Störsändare

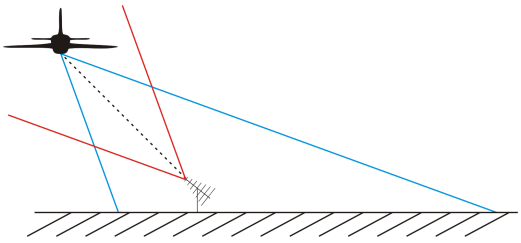
- När radarsystemet i programmet träder in i en störsändares respektive verkansområde antas radarsystemet vara i sändningsläge och störsändaren antas kunna börja störa omedelbart utan tidsfördröjning, dvs störsändaren svarar direkt på en mottagen radarpuls. På detta vis simuleras att störning alltid utförs vid rätt tidpunkt och ingen hänsyn till radarsystemets pulsrepetitionsfrekvens behöver tas. Resultatet blir att beräkningen av störövervikten i en viss punkt kan begränsas till att ske endast för de pixlar längs flygbanan som ligger i respektive störsändares verkansområde. För varje sådan pixel jämförs sedan de erhållna eko- och störsändareffekterna vilka i sin tur kan antas komma från en punktkälla av en pixels storlek.

- För att i en verklig situation uppnå bästa möjliga störverkan bör området som är avsett att maskeras inneslutas av störsändarna. Vidare bör störsändarnas verkansområde vara riktat bort ifrån skyddsområdet. I simuleringsprogrammet kommer dock de utplacerade störsändarna till en början vara riktade mot en gemensam front, dvs mot den tänkta flygbanan (Figur 3.3). Förenklingen är delvis en följd av att flygbanan alltid är den samma (föregående avsnitt), men huvudidéen bakom detta förfarande är att förenkla problemet med att räkna ut det snitt som uppstår mellan två eller flera samverkande störsändares effekter, se Figur 3.3.



Figur 3.3 *Störsändarna är riktade mot en gemensam front och i snittet adderas den utsända effekten.*

- I en verklig situation erhålls vidare en optimal störverkan om en störsändare kan följa störobjektets rörelser. Detta anses dock inte nödvändigt eftersom tillräckligt bra störverkan går att uppnå genom att i förväg tilta störsändarnas antenner så att respektive verkansområde koncentreras mot störobjektet. I några av de utförda störförsöken mot CARABAS har störsändarnas antenner haft en elevationsvinkel på 60° i förhållande till horisontalplanet i riktning mot den förväntade flygbanan. Med detta i åtanke kan det i programmet inbördes förhållandet mellan radar- och störsändarantenner förenklas genom att låta radarantennen vara riktad i exakt linje med riktningen på störsändarantennen. Det innebär att mottagare och sändare antas ligga i samma plan vid beräkningen av störövervikten, se Figur 3.4.



Figur 3.4 *Centrum på radarantennens huvudlob sammanfaller med centrum på störsändarens huvudlob.*

- Om störsändarna antas vara placerade i markhöjd kan träffytan från radarsystemets huvudlob användas som ett avgränsningsområde där användaren enkelt kan avläsa var störsändningen har gjort verkan.

3.4.4 Störövervikt

I uttrycket för störövervikt beskrivet i avsnitt 3.1 har en del justeringar genomförts som en följd av de förenklingar i omgivningens, radarsystemets och störsändarnas egenskaper enligt avsnitt 3.4.1, 3.4.2 respektive 3.4.3. I huvudsak handlar det om att eliminera de faktorer som på grund av dessa förenklingar ej längre anses relevanta.

- Eftersom det i simuleringsprogrammet inte förekommer några föremål för radarsystemet att detektera kan radarmålytan, σ, sättas lika med 1. Utan några punktmål antas radarsystemets utsända effekt komma från huvudlobens träffyta. Detta betyder att avståndet, R_r, mellan målet och radarantennen enbart användas i en riktning, dvs störövervikten beror av detta avstånd i kvadrat istället för en kvadrupel som tidigare.
- I och med användandet av radarsystemets antenndiagram kan även förlustfaktorn α för radarantennförstärkningen i respektive sidolob slopas. Förlustfaktorn α ingår på sätt och vis i antenndiagrammet.
- För att återspegla användandet av den procentuella störtiden (avsnitt 3.3.2) införs denna som en dämpningsfaktor, T, vilken ligger i intervallet [0, 1].
- Även en dämpningsfaktor har införts för att på något sätt representera markegenskaperna, se avsnitt 3.3.3. Markdämpningen, D, antages vara en konstant i intervallet [0, 1] oberoende av radarpulsens infallsvinkel. Ett D nära 0 beskriver stor dämpning av radarpulsens effekt, medan ett D nära 1 betyder att större delen av den utsända effekten reflekteras.

Det slutgiltiga uttrycket som används i simuleringsprogrammet för beräkning av störövervikten i en viss punkt får efter ovanstående förenklingar följande utseende:

$$J = \frac{P_j \cdot G_j \cdot \beta \cdot T}{4\pi \cdot R_j^2} \quad \text{och} \quad S = \frac{P_r \cdot G_r \cdot D}{4\pi \cdot R_r^2}.$$

När beräkningen av J och S i aktuell punkt är avklarad jämförs de båda effekterna för att avgöra vilken av den mottagna eko- eller störsändareffekt som skall återges grafiskt på skärmen.

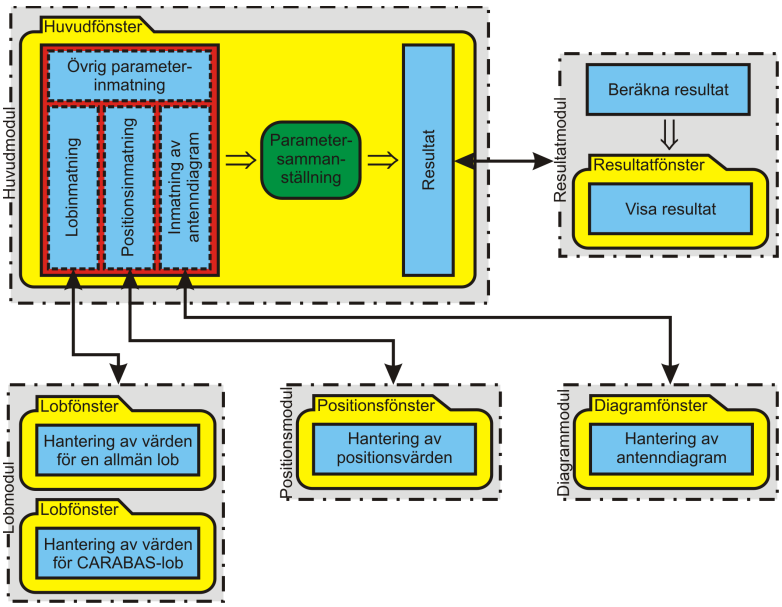
Värt att notera är att då det i simuleringen förekommer två eller fler störsändare kommer J att bli en summa av respektive störsändares uteffekt. Förutsättningen är förstås att den aktuella punkten längs flygbanan ligger inom störsändarnas verkansområde.

4 Implementering

Detta kapitel kommer översiktligt behandla implementeringen av en första prototyp av simuleringsprogrammet. För specifika programmeringslösningar hänvisas den intresserade läsaren till källkoden bifogad i bilaga B.1.

Källkoden till simuleringsprogrammet är skriven i ett för MatLab® 5 interpreterbart språk. Detta underlättar utvecklingsarbetet såtillvida att en specifik funktion kan plockas ut och enskilt testas utan att först behöva kompileras. En annan anledning till att MatLab® har använts som hjälpmedel vid utvecklingsarbetet är tillgången på beräkningsfunktioner och funktioner för grafiska gränssitt men framför allt för att stöd för kurvritning redan finns inbyggt.

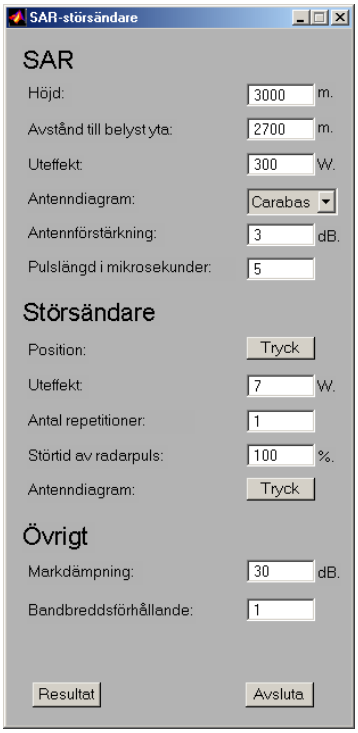
Den följande beskrivningen sker utifrån de moduler programmet består av. Med modul menas här en del av programmet som behandlar en specifik uppgift, t ex inläsning av störsändarnas position. De olika användargränssnitten i form av fönster har till stor del påverkat den valda moduluppdelningen, se Figur 4.1. Resultatmodulen beskrivs dock i kapitel 5.



Figur 4.1 *Principskiss av simuleringsprogrammets modulstruktur.*

4.1 Huvudfönster

När användaren startar upp simuleringsprogrammet möts denne av själva huvudfönstret där inmatning sker av de i avsnitt 3.3 beskrivna inparametrarna. Modulen fungerar som spindeln i nätet genom att samla in information om radarsystemet, störsändarna och de övriga egenskaperna som sedan skall skickas vidare för bearbetning. Viss grundläggande felhantering har lagts in för att undvika programkrasch vid felaktiga beräkningstillstånd, t ex division med noll etc.



Figur 4.2 *Huvudfönstret fördelar arbetet på de övriga modulerna.*

Utifrån huvudfönstret (Figur 4.2) sker kommunikation med de andra modulerna/fönstrena beroende på vad användaren för tillfället önskar göra. På de ställen i huvudfönstret där inmatning inte kan göras direkt överlåts uppgiften till en annan modul. Att på det här viset frikoppla moduler underlättar då programmet skall byggas om eller modifieras. En modul kan enkelt läggas till eller tas bort och förändringar kan skötas på respektive modulnivå.

Efter inläsning lagras varje inparameter i en gemensam inmatningspost vilken vid behov skickas till resultatmodulen där beräkning av själva störövervikten sker. Men för att undvika att användaren måste knappa in alla inparametrarna från början varje gång en mindre ändring ska göras, har varje inparameter tilldelats ett i förväg definierat värde. De fördefinierade värdena är de standardvärden som beskrivits i avsnitt 3.3.1, 3.3.2 samt 3.3.3. Detta innebär bl a att radarantennen automatiskt får CARABAS-egenskaper vid uppstart av simuleringsprogrammet.

4.2 Huvudlobsutformning

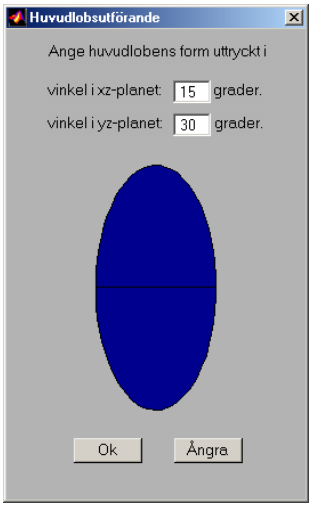
På grund av tidsbrist har inte stöd för inläsning av ett radarsystems antenndiagram implementerats, utan förenklingen beskriven under den första punkten i avsnitt 3.4.2 används istället. Eftersom CARABAS-antennen har en azimutvinkel på 90° och uteffekten till en början anses vara konstant innanför huvudloben så kan denna egenskap föras över på störsändarna. Dessa får då ett antenndiagram som sträcker sig från -45° till 45° i störriktningen.

I ett försök att även kunna hantera en allmän huvudlob har dock två användargränssnitt för inmatning av lobens utförande införts . Valet av huvudlob sker i huvudfönstret och beroende av val kommer beräkningarna av störövervikten för respektive lob att skilja sig något åt.

4.2.1 Allmän huvudlob

Fördelen med att kunna använda sig av en allmän huvudlob är att även andra radarsystem än just CARABAS till viss del kan simuleras.

I inmatningsfönstret (Figur 4.3) visas formen på det avgränsningsområde som uppstår då huvudloben projiceras längs lodlinjen mot horisontalplanet (xy-planet). Formen på den projicerade ytan bestäms av huvudlobens (konens) vinkel θ i xz-planet och vinkeln ϕ i yz-planet.

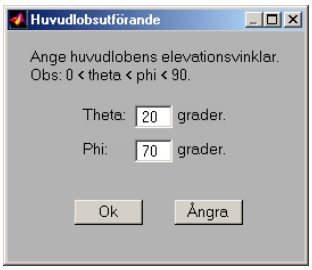


Figur 4.3 *En allmän huvudlobs avgränsningsområde projicerad längs lodlinjen.*

Viktigt att påpeka är att användandet av en allmän huvudlob bör användas med viss försiktighet eftersom vinklarna som bestämmer huvudlobens utformning i nuvarande version kan väljas godtyckligt stora. Det problem som kan uppstå med en för stor lob i kombination med en för stor elevationsvinkel är att loben hamnar ovanför horisontalplanet. Följden av detta är att ett oändligt stort avgränsningsområde genereras vilket simuleringsprogrammet för tillfället inte kan behandla.

4.2.2 CARABAS-huvudlob

Jämfört med en allmän huvudlob så är det i CARABAS-fallet lättare att kontrollera användarens inmatningar. Men för att undvika de problem som uppstår i samband med oändliga avgränsningsområden har de tillåtna vinklarna som bestämmer huvudloben i elevationsled lagts i intervallet $0° < \theta < \phi < 90°$.

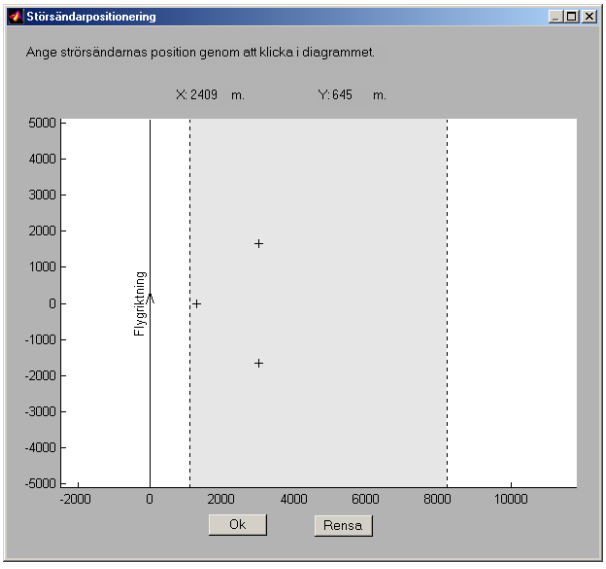


Figur 4.4 *I CARABAS-fallet bestäms bredden på avgränsningsområdet av vinklarna θ och ϕ.*

I inmatningsfönstret (Figur 4.4) ges de värden på vinklarna θ och ϕ som bestämmer CARABAS karaktäristiska elevationsvinkel. Värt att notera är att vinklarna för bestämmandet av huvudloben i CARABAS-fallet inte har samma innebörd som i det allmänna fallet. Gemensamt för de båda fallen är dock att huvudlobens utformning avgör hur det avgränsade området i horisontalplanet kommer att se ut.

4.3 Positionering av störsändare

I positioneringsfönstret (Figur 4.5) räknas en störsändares aktuella position ut med hjälp av ett koordinatsystem vilket är fixerat vid x- respektive y-axeln. Vid y-axeln läggs flygbanan in, längs vilken radarsystemet förflyttar sig i positiv led. Längs x-axeln beräknas det horisontella avståndet mellan störsändare och radarsystem.



Figur 4.5 *Tre störsändare är utplacerade och en fjärde är på väg att placeras i position (2409, 645).*

Utifrån detta referenssystem placeras sedan störsändare representerade av ”+”-tecken ut. För att underlätta utplaceringen av störsändarna har markören försetts med ett koordinatpar vilket visar markörens nuvarande position.

Formen på radarantennens huvudlob avgör bredden på avgränsningsområdet vilket i positioneringsfönstret markeras av det grå fältet. Eftersom det i nuvarande version inte går att markera ett tänkt skyddsområde får användaren ungefärligt uppskatta skyddsområdets bredd respektive djup för att sedan med hjälp av avgränsningsområdet och koordinatssystemet avgöra var de olika störsändarna bäst skall placeras.

För att användaren inte skall behöva lägga in nya störsändare efter varje simulering sparas den aktuella positioneringen tills användaren själv väljer att rensa och börja om. Detta underlättar då en viss störformation skall testas med olika värden på radarsystem, störsändare och omgivning.

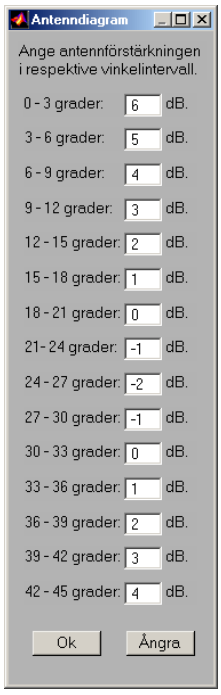
4.4 Antenndiagram för störsändare

I fönstret där antenndiagrammet matas in (Figur 4.6) behöver bara hälften av värdena anges. Anledningen till detta är att störsändarens antenndiagram antas vara symmetriskt kring den störriktning mot vilken antennen är riktad.

På grund av förenklingarna beskrivna i avsnitt 3.4 är antenndiagrammet för en störsändare i nuvarande version begränsat till att gälla från -45° till 45° i förhållande till störriktningen, denna uppdelning blir således även störsändarens verkansområde. Vidare är diagrammet indelat i mindre vinkelintervall inom vilken en viss antennförstärkning antas vara konstant. Med konstant menas här att radarsystemet tar emot samma störeffekt inom ett delintervall förutsatt att avståndet till störsändaren är oförändrat.

För varje delintervall i inmatningsfönstret anges antennförstärkningen i antal decibel, dB, där varje 3 dB motsvarar en fördubbling av uteffekten medan varje -3 dB utgör en effekthalvering.

Även här kommer de inmatade värdena att sparas för att underlätta för en användare när denne vill utföra en serie simuleringar med samma typ av störsändare.



Figur 4.6
Förstärkningen anges i decibel.

5 Resultat

Eftersom uppgiften för examensarbetet har varit att modellera och programmera ett simuleringssystem så blev utvecklingsarbetet en iterativ process där problemlösande och implementering vävs samman tills ett tillfredsställande resultat är uppnått. Med andra ord; processen pågår tills ett *tillräckligt* bra resultat har presterats. Vad som sedan anses vara tillräckligt är mer ett ställningstagande till modellens detaljnivå kontra tidstillgång. En mer noggrann modell kräver mer tankearbete och därmed mer tid. Med de i avsnitt 3.4 beskrivna förenklingarna anses dock resultatet vara av tillräckligt god kvalitet.

5.1 Tolkning av resultatet

I avsnitt 3.2 beskrivs det önskade resultatet som att programmet skall beräkna och på något vis grafiskt presentera styrkan hos störövervikten. Det som visas i resultaten nedan är dock inte störövervikten utan den starkaste mottagna effekten, dvs eko- eller störsändareffekt, representerad med olika färger. För varje punkt längs flygbanan som ligger inom en störsändares verkansområde sammanställs och visas den totala mottagna störsändareffekten om denna är större än den mottagna ekoeffekten. För att se styrkefördelningen på de mottagna störsändareffekterna har dessa givits en RGB-kombination, där R = röd motsvarar den starkaste störsändareffekten och B = blå motsvarar den svagaste störsändareffekten. Skulle den mottagna störsändareffekten understiga den mottagna ekoeffekten visas ingenting i det område där detta inträffar.

Det hyperbellika utseendet på störövervikten beror på att radarsystemet registrerar ett eko som om det hade kommit ifrån den riktning i vilken radarantennen är riktad, dvs vinkelrätt mot färdriktningen. Detta betyder att när radarsystemet träder in i en störsändares verkansområde kommer all den mottagna störsändareffekten, oavsett riktning, att uppfattas som om den kom ifrån en punkt tvärs flygriktningen. Att färgfältet breder ut sig bakom själva störsändaren beror på att störsändaren börjar störa omedelbart när en radarpuls tas emot.

I syfte att skapa en bättre överblick av det aktuella området har skalan i efterföljande diagram dubblerats i jämförelse med det ursprungliga posi-

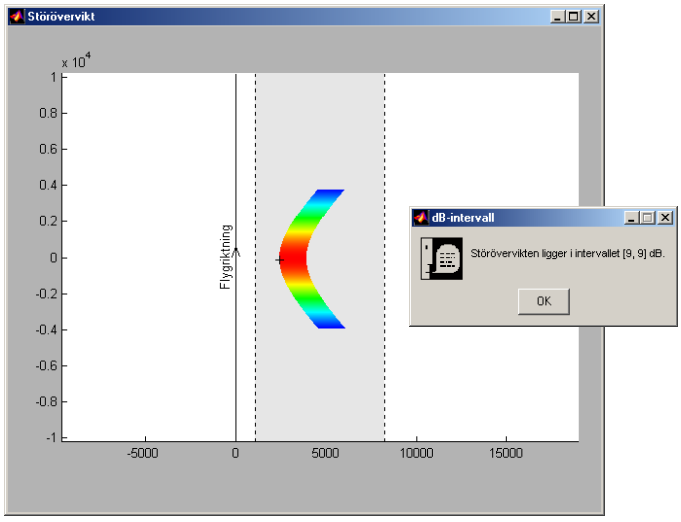
tioneringsområdet beskrivet i avsnitt 4.3. Avstånden återgivna längs x- respektive y-axeln är dock de samma och skall tolkas som antalet meter från en tänkt mittpunkt i horisontalplanet längs den utritade flygbanan. Vidare skall den gråtonade ytan i bakgrunden på resultatdiagrammet ses som det av radarsystemet avsökta området, dvs huvudlobens träffyta längs flygbanan, jfr avsnitt 4.3.

För att bättre kunna se skillnaden på olika störsituationer har de i avsnitt 3.3.1 beskrivna CARABAS-specifika egenskaperna använts vid simuleringarna. Övriga egenskaper av intresse återfinns i Figur 4.2. Vid simuleringen beskriven i avsnitt 5.5 har dock vissa ändringar gjorts för att tydligare åskådliggöra vad som händer då den mottagna störsändareffekten är otillräcklig.

För övrigt informeras användaren via ett litet dialogfönster om vilket intervall den mottagna störeffekten ligger i. Som nämndes i avsnitt 3.3.2 bör störövervikten uppgå till ett par decibel för att tillräcklig störverkan skall uppnås.

5.2 En störsändare med konstant förstärkning

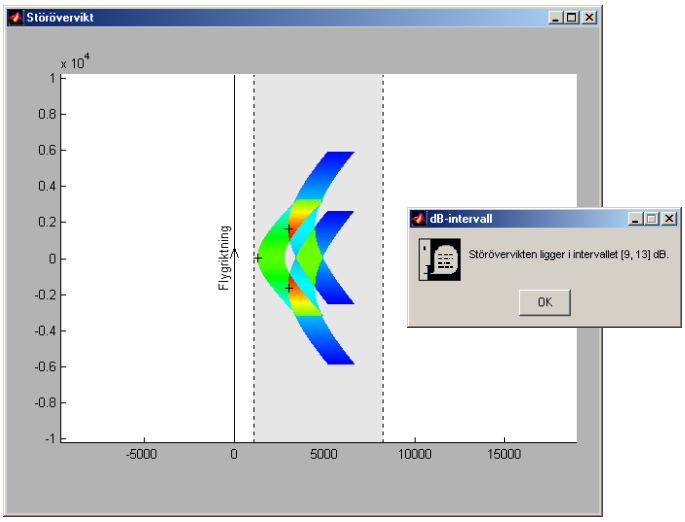
I Figur 5.1 nedan ses en störsändare med ett antenndiagram vars delförstärkningar är de samma i samtliga riktningar. Att figuren skiftar färg från rött till blått trots den konstanta antennförstärkningen beror på att den mottagna störsändareffekten har ett omvänt avståndsberoende.



Figur 5.1 *En störsändare med konstant antennförstärkning genererar störst effekt (rött) på det kortaste avståndet och minst effekt (blått) på det längsta avståndet.*

5.3 Flera störsändare med konstant förstärkning

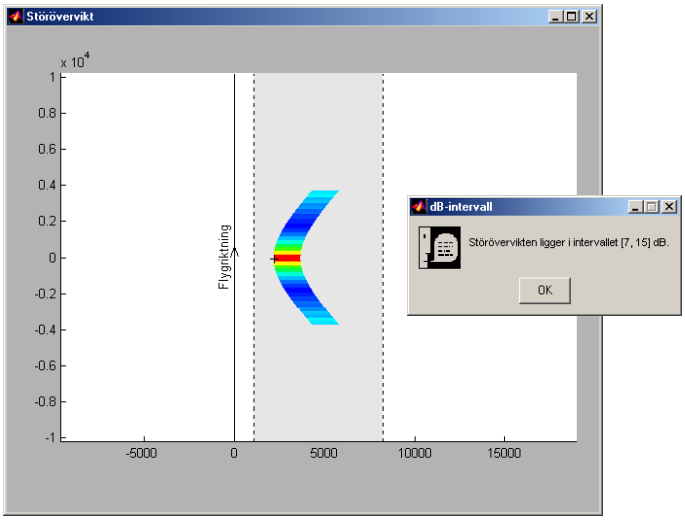
Vid denna simulering har tre störsändare placerats ut för att visa hur dessa samverkar. I resultatdiagrammet i Figur 5.2 nedan återges störsändarnas sammanlagda effekter som ett snitt i färgbanden där dessa överlappar varandra.



Figur 5.2 *Då antennförstärkningen är konstant erhålls den största störverkan i snitten mellan störsändarnas respektive verkansområden.*

5.4 En störsändare med variabel förstärkning

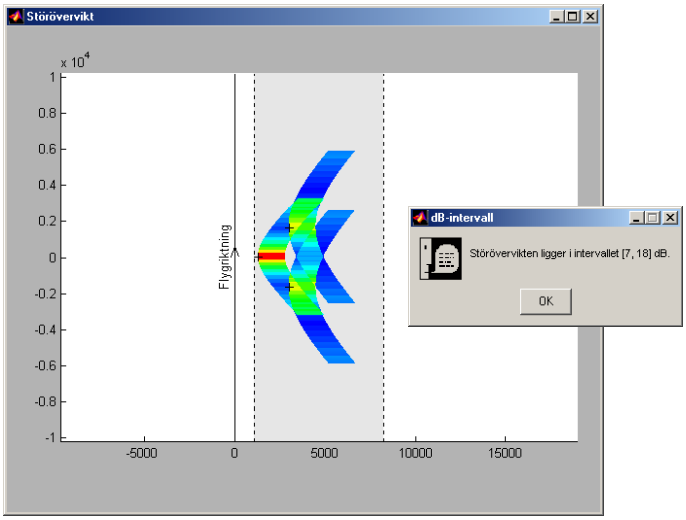
Notera minskningen (blått område) av störeffekten i Figur 5.3 nedan som uppstår då antenndiagrammet från Figur 4.6 används, jfr Figur 5.1.



Figur 5.3 *Antenndiagrammets vinkelintervall framträder som skikt i den utritade störeffekten.*

5.5 Flera störsändare med variabel förstärkning

För att enklare kunna göra en jämförelse med simuleringen beskriven i avsnitt 5.3 har samma störsändarformering och antenndiagram använts. Resultatet av simuleringen återfinns i Figur 5.4 nedan.

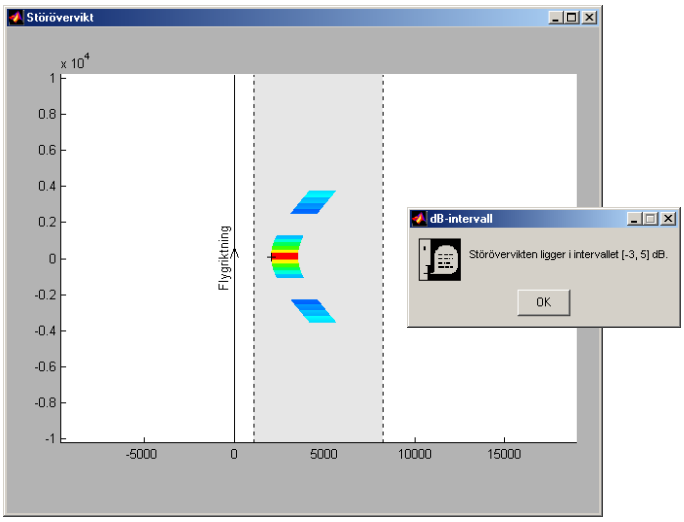


Figur 5.4 *Störeffekten från störsändaren närmast flygbanan har fått en jämnare fördelning tack vare samverkan från de båda andra störsändarna.*

Observera att den starkaste störeffekten fortfarande återfinns på det kortaste avståndet, men tack vare störsändarnas samverkan har den totala störeffekten i det gemensamma verkansområdet jämnats ut och expanderat.

5.6 En störsändare med otillräcklig störeffekt

I denna simulering har vissa av inparametrarna ändrats i syfte att illustrera vad som händer då den mottagna störsändareffekten är mindre än den mottagna ekoeffekten. Samma antenndiagram och positionering som i simuleringen beskriven i avsnitt 5.4 har använts. Notera ”hålen” i den utritade störeffekten återgiven i Figur 5.5. Dessa hål uppkommer som en följd av att antennförstärkningen inte är tillräckligt stor inom detta vinkelintervall i störsändarens antenndiagram. I dialogrutan ses även att det aktuella intervallet för störövervikten inkluderar värden mindre än 0 dB.



Figur 5.5 *I de områden där störeffekten är mindre än ekoeffekten uppstår "hål" inom vilka störövervikten är mindre än 0 dB.*

Jämfört med föregående simuleringar har nedanstående parameterförändringar gjorts för att framkalla ovan beskrivna fenomen.

- Radarsystemets antennförstärkning inom huvudloben har höjts från 3 till 5 dB.
- Bandbreddsförhållandet har halverats från 1 till 0.5. Eftersom viss marginal önskas i störpulsens bandbredd, bidrar denna förändring till att simuleringen blir mer realistisk.
- Den procentuella störtiden för störsändaren har satts till 50%.
- Störsändarens uteffekt har minskats från 7 till 5 W.

6 Avslutning

I det här kapitlet summeras arbetet genom att kommentera de slutsatser som dragits från utvecklingsarbetet, redogöra för kända fel samt presentera förslag till framtida förbättringar.

6.1 Slutsatser

En slutsats som direkt kan dras ur ett modelleringsarbete av detta slag är att det krävs god kännedom om vilken noggrannhet som önskas i det förväntade resultatet. Ju högre noggrannhet som önskas desto högre detaljnivå krävs i modelleringen, vilken följaktligen blir mer komplicerad. I sitt nuvarande tillstånd fungerar simuleringsprogrammet tillfredställande eftersom de förenklingar som gjorts anses acceptabla för presentation av relevanta störsituationer.

Vidare kan sägas att användandet av olika färgkombinationer för representation av mottagen störeffekt var ett synnerligen lyckat val eftersom ytterligare en dimension kan tillföras det redan tvådimensionella presentationsdiagrammet.

6.2 Kända fel

Eftersom vinklarna som bestämmer formen på den allmänna huvudloben kan göras godtyckligt stora kan lobens avgränsningsområde delvis komma att hamna ovanför horisontalplanet. Detta resulterar i att den del av avgränsningsområdet som hamnar i horisontalplanet blir oändligt stort och presentationen av störövervikten blir meningslös. I dagsläget finns det i simuleringsprogrammet ingen modell som kan handskas med ett sådant fall.

Orsaken till att problemet kvarstår är att det anses mer viktigt att kunna göra en mer generell simulering med en allmän huvudlob än att tillhandahålla en fullständig felhantering. En normal simulering går dock utmärkt att genomföra förutsatt att användaren har någorlunda god kunskap om de olika inparametrarna. Med en normal simulering menas att inparametrarna antar värden vilka överensstämmer med verkligheten. Som exempel kan nämnas att en parabolantenn har en huvudlob vars lobvinklar maximalt bara är några tiotals grader.

6.3 Framtida förbättringar

Under arbetets gång har nya intressanta lösningar och förbättringar upptäckts, men eftersom tiden för examensarbetet är begränsad är det en omöjlighet att hinna med allt. Här nedan ges dock några förslag på möjliga framtida förbättringar.

6.3.1 Utvecklingshjälpmedel

I dagsläget används MatLab® 5 för att enkelt kunna rita ut grafiska objekt, göra besvärliga beräkningar samt göra lättförståeliga användargränssnitt. Nackdelen med att skriva program i ett interpretativt språk som MatLab® är att stora beräkningsintensiva operationer, exempelvis stora for-loopar, exekverar väldigt långsamt. I en framtida version kan eventuellt koden tänkas kompileras till en exekverbar binärfil vilket enligt dr Gunnar Hillerström (FOI) uppskattningsvis skulle snabba upp beräkningsarbetet med åtminstone en faktor tio.

6.3.2 Omgivning

För att på ett mer realistiskt sätt kunna efterlikna ett målområde skulle en karta kunna läsas in, skalas och anpassas. I databaser på Internet och även i speciella kartprogram finns tillgång till detaljerade kartor som kan verka som underlag vid en störsimulering. Med hjälp av en karta blir överblicken tydligare genom att ett tänkt skyddsområdet enklare kan lokaliseras. En annan viktig fördel med användandet av skalenliga kartor är att ett enhetligt referenssystem erhålls i form av Lat-Long positionering.

I en verklig situation är det relativt osannolikt att radarsystemets flygbana är känd i förväg. Därför bör en godtycklig flygbana kunna placeras in i området. Vidare är det önskvärt att användaren på egen hand skall kunna markera ut ett aktuellt skyddsområde direkt i simuleringsprogrammet.

Värt att nämna är att viss möjlighet att kunna placera in objekt som kan detekteras av radarsystemet skulle verklighetsanpassa simuleringarna ytterligare. Att implementera en modell för detta ändamål kräver dock mer ingående studier i antennteori, material- och vågutbredningslära än vad som ryms inom tidsramen för ett examensarbete.

6.3.3 Radar

I en framtida version av simuleringsprogrammet är det tänkbart att bygga in funktion för inläsning av antenndiagram, dels på liknande sätt som för störsändare men även från fil. Att förlägga ett antenndiagrams värden på

fil skulle snabba upp hanteringen av byte mellan radarsystem av olika typ.

Genom att införa ett antenndiagram som mer precist definierar radarlobernas utseende kan även den nuvarande uppdelningen av lober slopas och därmed kan problemen med oändliga avgränsningsområden elimineras.

6.3.4 Störsändare

För att ytterliggare förbättra trovärdigheten i störsimuleringen bör användaren kunna rikta och tilta störsändarnas antenner efter eget tycke. Detta i kombination med en godtycklig flygbana skulle erbjuda möjlighet att simulera en störsituation där ett skyddsområde är ”inhägnat” av störsändare.

Även här skulle det vara önskvärt ha möjlighet till inläsning av en störsändares antenndiagram från fil.

6.3.5 Störövervikt

Om det finns objekt att detektera i omgivningen behöver ett värde för dess effektiva radarmålyta inkluderas i uttrycket för störövervikt. I simuleringsprogrammets nuvarande form har denna faktor eliminerats i uttrycket för störövervikt.

Litteraturförteckning

Hackman, Peter (1997), *Boken med kossan på*, trettonde upplagan. Hut, Hyfs och Hållning Productions, Linköping.

Hyberg, Per (1993), *Radarmotmedelsteknik*. Försvarets forskningsanstalt, Linköping. A 10052-1.1.

Jernemalm, Veine (1988), *Aktiv störteknik – Radar*. Försvarets forskningsanstalt, Linköping. C 30506-3.6.

Josefsson, Lars & Berggren, Göran (1997), *Telekrig – lärobok för armén.* Försvarsmakten, Stockholm. M7746-168001.

Kingsley, Simon & Quegan, Shaun (1992), *Understanding radar systems*. McGraw-Hill, Maidenhead. ISBN 0-07-707426-2.

Levanon, Nadav (1988), *Radar principles*. John Wiley & Sons, New York. ISBN 0-471-85881-1.

Littke, Ann Kathrine & Sundström, Olle (1995), *Försvarets forskningsanstalt 1945-1995.* PROBUS Förlag HB, Stockholm. ISBN 91-87184-39-7.

Merkel, Magnus (1996), *Tekniska Rapporter och examensarbeten*. Linköpings Universitet.

Persson, Arne & Böiers, Lars-Christer (1996), *Analys i flera variabler*, andra upplagan. Studentlitteratur, Lund. ISBN 91-44-26921-8.

Pärt-Enander, Eva & Sjöberg, Anders (1998), *Användarhandledning för MatLab® 5*. Uppsala Universitet. ISBN 91-506-1326-X.

Sandia National Laboratories (2001), *Synthetic Aperture Radar* [Online]. Tillgänglig på http://www.sandia.gov/RADAR/sar.html [Besökt 18 jan -01].

Stimson, George W (1998), *Introduction to Airborne Radar*, second edition. SciTech Publishing Inc., Mendham. ISBN 1-891121-01-4.

Bilagor

B.1 Källkod

Här återfinns de MatLab®-filer som ingår i applikationen SimSAR vilken simulerar förhållandet mellan ett SAR-system och ett antal störsändare.

SimSAR.m

```
function SimSAR()

default.height = 3000;
default.range = 2700;
default.rpower = 300;
default.choice = 2;
default.generalmainlobe = [15, 30];
default.carabasmainlobe = [20, 70];
default.radargain = 3;
default.pulseduration = 5;
default.jammerpositions = [];
default.jpower = 7;
default.repetitions = 1;
default.jamtime = 100;
default.jammergain = [6,5,4,3,2,1,0,-1,-2,-1,0,1,2,3,4];
default.groundgain = 30;
default.bandratio = 1;

GeneralInputs(default);
disp('Done!');
%end SimSAR
```

GeneralInputs.m

```
function input = GeneralInputs(default)

startptr = StartWindow;
set(0, 'ShowHiddenHandles', 'on'); %Make all objects available.
set(startptr, 'Resize', 'off');
set(startptr, 'UserData', default);
SetDefaultValues(startptr, default);
uiwait(startptr); %Halt execution until user is done inputing
values.
if (ishandle(startptr))
   close(startptr);
else
   disp('aborted');
end
%end GeneralInputs
```

```
function SetDefaultValues(startptr, default)
objptr = findobj(startptr, 'Tag', 'Height');
set(objptr, 'String', num2str(default.height));
set(objptr, 'UserData', default.height);

objptr = findobj(startptr, 'Tag', 'Range');
set(objptr, 'String', num2str(default.range));
set(objptr, 'UserData', default.range);

objptr = findobj(startptr, 'Tag', 'RadarPower');
set(objptr, 'String', num2str(default.rpower));
set(objptr, 'UserData', default.rpower);

objptr = findobj(startptr, 'Tag', 'RadarMainLobe');
choice = default.choice;
switch(choice)
case 1
   set(objptr, 'UserData', default.generalmainlobe);
case 2
   set(objptr, 'UserData', default.carabasmainlobe);
end
set(objptr, 'Value', choice);

objptr = findobj(startptr, 'Tag', 'RadarGain');
set(objptr, 'String', num2str(default.radargain));
set(objptr, 'UserData', default.radargain);

objptr = findobj(startptr, 'Tag', 'PulseDuration');
set(objptr, 'String', num2str(default.pulseduration));
set(objptr, 'UserData', default.pulseduration);

objptr = findobj(startptr, 'Tag', 'Position');
set(objptr, 'UserData', default.jammerpositions);

objptr = findobj(startptr, 'Tag', 'JammerPower');
set(objptr, 'String', num2str(default.jpower));
set(objptr, 'UserData', default.jpower);

objptr = findobj(startptr, 'Tag', 'Repetitions');
set(objptr, 'String', num2str(default.repetitions));
set(objptr, 'UserData', default.repetitions);

objptr = findobj(startptr, 'Tag', 'JamTime');
set(objptr, 'String', num2str(default.jamtime));
set(objptr, 'UserData', default.jamtime);

objptr = findobj(startptr, 'Tag', 'JammerGain');
set(objptr, 'UserData', default.jammergain);

objptr = findobj(startptr, 'Tag', 'GroundGain');
set(objptr, 'String', num2str(default.groundgain));
set(objptr, 'UserData', default.groundgain);

objptr = findobj(startptr, 'Tag', 'BandRatio');
set(objptr, 'String', num2str(default.bandratio));
set(objptr, 'UserData', default.bandratio);
%end SetDefaultValues
```

StartCode.m

```
function StartCode(action)

default = get(gcbf, 'UserData');
switch(action)
case 'height'
   str = get(gcbo, 'String');
   height = StrCheck(str, 500, 10000);
   if (height == 'err')
      set(gcbo, 'String', num2str(default.height));
      set(gcbo, 'UserData', default.height);
   else
      set(gcbo, 'UserData', height);
   end
case 'range'
   str = get(gcbo, 'String');
   range = StrCheck(str, 0, 10000);
   if (range == 'err')
      set(gcbo, 'String', num2str(default.range));
      set(gcbo, 'UserData', default.range);
   else
      set(gcbo, 'UserData', range);
   end
case 'radarpower'
   str = get(gcbo, 'String');
   rpower = StrCheck(str, 0, 1000);
   if (rpower == 'err')
      set(gcbo, 'String', num2str(default.rpower));
      set(gcbo, 'UserData', default.rpower);
   else
      set(gcbo, 'UserData', rpower);
   end
case 'radarmainlobe'
   angles = MainLobeAngles;
   set(gcbo, 'UserData', angles);
case 'radargain'
   str = get(gcbo, 'String');
   radargain = StrCheck(str, 0, 50);
   if (radargain == 'err')
      set(gcbo, 'String', num2str(default.radargain));
      set(gcbo, 'UserData', default.radargain);
   else
      set(gcbo, 'UserData', radargain);
   end
case 'pulseduration' %Given in microseconds.
   str = get(gcbo, 'String');
   pduration = StrCheck(str, 0, 10);
   if (pduration == 'err')
      set(gcbo, 'String', num2str(default.pulseduration));
      set(gcbo, 'UserData', default.pulseduration);
   else
      set(gcbo, 'UserData', pduration);
   end
case 'position'
   positions = JammerPositions;
   set(gcbo, 'UserData', positions);
case 'jammerpower'
```

```
    str = get(gcbo, 'String');
    jpower = StrCheck(str, 0, 50);
    if (jpower == 'err')
       set(gcbo, 'String', num2str(default.jpower));
       set(gcbo, 'UserData', default.jpower);
    else
       set(gcbo, 'UserData', jpower);
    end
case 'repetitions'
    str = get(gcbo, 'String');
    repetitions = StrCheck(str, 1, 10);
    if (repetitions == 'err')
       set(gcbo, 'String', num2str(default.repetitions));
       set(gcbo, 'UserData', default.repetitions);
    else
       set(gcbo, 'UserData', repetitions);
    end
case 'jamtime'
    str = get(gcbo, 'String');
    jamtime = StrCheck(str, 0, 100);
    if (jamtime == 'err')
       set(gcbo, 'String', num2str(default.jamtime));
       set(gcbo, 'UserData', default.jamtime);
    else
       set(gcbo, 'UserData', jamtime);
    end
case 'jammergain'
    gain = JammerGain;
    set(gcbo, 'UserData', gain);
case 'groundgain'
    str = get(gcbo, 'String');
    groundgain = StrCheck(str, 20, 70);
    if (groundgain == 'err')
       set(gcbo, 'String', num2str(default.groundgain));
       set(gcbo, 'UserData', default.groundgain);
    else
       set(gcbo, 'UserData', groundgain);
    end
case 'bandratio'
    str = get(gcbo, 'String');
    bandratio = StrCheck(str, 0, 1);
    if (bandratio == 'err')
       set(gcbo, 'String', num2str(default.bandratio));
       set(gcbo, 'UserData', default.bandratio);
    else
       set(gcbo, 'UserData', bandratio);
    end
case 'result'
    ShowResult(CollectInputs);
case 'exit'
    uiresume(gcbf);
end
%end StartCode
```

```
function num = StrCheck(str, min, max)
if ((str == 'i') | (str == 'j')) %Eliminate imaginary values.
   temp = [];
else
   temp = str2num(str);
end

if (isempty(temp) | (temp < min) | (temp > max))
   errptr = errordlg(['Värdet måste ligga i intervallet [',
int2str(min), ', ', int2str(max), ']!'], 'Ett fel uppstod');
   set(errptr, 'WindowStyle', 'modal');
   num = 'err';
else
   num = temp;
end
%end StrCheck



function values = CollectInputs
startptr = findobj(0, 'Tag', 'StartWindow');

objectptr = findobj(startptr, 'Tag', 'Height');
temp.height = get(objectptr, 'UserData');

objectptr = findobj(startptr, 'Tag', 'Range');
temp.range = get(objectptr, 'UserData');

objectptr = findobj(startptr, 'Tag', 'RadarPower');
temp.radarpower = get(objectptr, 'UserData');

objectptr = findobj(startptr, 'Tag', 'RadarMainLobe');
choice = get(objectptr, 'Value');
angles = get(objectptr, 'UserData');
temp.radarlobe = [choice, angles];

objectptr = findobj(startptr, 'Tag', 'RadarGain');
temp.radargain = get(objectptr, 'UserData');

objectptr = findobj(startptr, 'Tag', 'PulseDuration');
temp.pulseduration = get(objectptr, 'UserData');

objectptr = findobj(startptr, 'Tag', 'Position');
temp.jammerpositions = get(objectptr, 'UserData');

objectptr = findobj(startptr, 'Tag', 'JammerPower');
temp.jammerpower = get(objectptr, 'UserData');

objectptr = findobj(startptr, 'Tag', 'Repetitions');
temp.repetitions = get(objectptr, 'UserData');

objectptr = findobj(startptr, 'Tag', 'JamTime');
temp.jamtime = get(objectptr, 'UserData');

objectptr = findobj(startptr, 'Tag', 'JammerGain');
temp.jammergain = get(objectptr, 'UserData');

objectptr = findobj(startptr, 'Tag', 'GroundGain');
```

```
temp.groundgain = get(objectptr, 'UserData');

objectptr = findobj(startptr, 'Tag', 'BandRatio');
temp.bandratio = get(objectptr, 'UserData');

values = temp;
%end CollectInputs
```

MainLobeAngles.m

```
function angles = MainLobeAngles()

startptr = findobj(0, 'Tag', 'StartWindow');
default = get(startptr, 'UserData');
fieldptr = findobj(gcbf, 'Tag', 'RadarMainLobe');
choice = get(fieldptr, 'Value');
switch(choice)
case 1
   MLAWindow; %MLA = Main Lobe Angles.
   mlaptr = findobj(0, 'Tag', 'MLAWindow');
   set(mlaptr, 'WindowStyle', 'modal');
   set(mlaptr, 'UserData', default.generalmainlobe);
   theta = default.generalmainlobe(1)*pi/180; %Use default
angles to start with.
   phi = default.generalmainlobe(2)*pi/180;

   fieldptr = findobj(mlaptr, 'Tag', 'Theta'); %Update working
values.
   set(fieldptr, 'String',
num2str(default.generalmainlobe(1)));
   set(fieldptr, 'UserData', default.generalmainlobe(1));
   fieldptr = findobj(mlaptr, 'Tag', 'Phi');
   set(fieldptr, 'String',
num2str(default.generalmainlobe(2)));
   set(fieldptr, 'UserData', default.generalmainlobe(2));

   c = (tan(phi/2))/(tan(theta/2));
   t = 0:0.1:2*pi;
   if (theta > phi)
      area(cos(t), c*sin(t));
   else
      area(cos(t)./c, sin(t));
   end
   axis('equal');
   axis('off');

   uiwait(mlaptr); %Halt execution untill user is done inputing
values.
   if (ishandle(mlaptr)) %If not aborted, collect input angles.
      fieldptr = findobj(mlaptr, 'Tag', 'Theta');
      theta = get(fieldptr, 'UserData');
      fieldptr = findobj(mlaptr, 'Tag', 'Phi');
      phi = get(fieldptr, 'UserData');
      close(mlaptr);
   else
      theta = default.generalmainlobe(1);
      phi = default.generalmainlobe(2);
    end
```

```
case 2
   MLEWindow; %MLE = Main Lobe Elevation.
   mleptr = findobj(0, 'Tag', 'MLEWindow');
   set(mleptr, 'WindowStyle', 'modal');
   set(mleptr, 'UserData', default.carabasmainlobe);
   theta = default.carabasmainlobe(1)*pi/180;
   phi = default.carabasmainlobe(2)*pi/180;

   fieldptr = findobj(mleptr, 'Tag', 'Theta');
   set(fieldptr, 'String',
num2str(default.carabasmainlobe(1)));
   set(fieldptr, 'UserData', default.carabasmainlobe(1));
   fieldptr = findobj(mleptr, 'Tag', 'Phi');
   set(fieldptr, 'String',
num2str(default.carabasmainlobe(2)));
   set(fieldptr, 'UserData', default.carabasmainlobe(2));

   uiwait(mleptr);
   if (ishandle(mleptr))
      fieldptr = findobj(mleptr, 'Tag', 'Theta');
      theta = get(fieldptr, 'UserData');
      fieldptr = findobj(mleptr, 'Tag', 'Phi');
      phi = get(fieldptr, 'UserData');
      close(mleptr);
   else
      theta = default.carabasmainlobe(1);
      phi = default.carabasmainlobe(2);
   end
end

angles = [theta, phi]; %Angles are given in degrees.
%end MainLobeAngles
```

MLACode.m

```
function MLACode(action)

set(0, 'ShowHiddenHandles', 'on');
startptr = findobj(0, 'Tag', 'StartWindow');
mlaptr = findobj(0, 'Tag', 'MLAWindow');

switch(action)
case 'theta'
   str = get(gcbo, 'String');
   temp = StrCheck(str);
   if (temp == 'err')
      default = get(gcbf, 'UserData'); %Input incorrect -> use
default value.
      set(gcbo, 'String', num2str(default(1))); %Update text
field.
      set(gcbo, 'UserData', default(1)); %Update working value.
      theta = default(1)*pi/180;
   else
      set(gcbo, 'UserData', temp); %Update working value.
      theta = temp*pi/180;
   end

   fieldptr = findobj(mlaptr, 'Tag', 'Phi');
```

```
   phi = get(fieldptr, 'UserData')*pi/180;

   c = (tan(phi/2))/(tan(theta/2));
   t = 0:0.1:2*pi;
   figure(mlaptr); %Select correct figure to draw object in.
   if (theta > phi)
      area(cos(t), c*sin(t));
   else
      area(cos(t)./c, sin(t));
   end
   axis('equal');
   axis('off');
case 'phi' %See comments for 'theta' above.
   str = get(gcbo, 'String');
   temp = StrCheck(str);
   if (temp == 'err')
      default = get(gcbf, 'UserData');
      set(gcbo, 'String', num2str(default(2)));
      set(gcbo, 'UserData', default(2));
      phi = default(2)*pi/180;
   else
      set(gcbo, 'UserData', temp);
      phi = temp*pi/180;
   end

   fieldptr = findobj(mlaptr, 'Tag', 'Theta');
   theta = get(fieldptr, 'UserData')*pi/180;

   c = (tan(phi/2))/(tan(theta/2));
   t = 0:0.1:2*pi;
   figure(mlaptr);
   if (theta > phi)
      area(cos(t), c*sin(t));
   else
      area(cos(t)./c, sin(t));
   end
   axis('equal');
   axis('off');
case 'ok'
   uiresume(mlaptr);
case 'cancel'
   default = get(gcbf, 'UserData'); %Restore to default values.
   fieldptr = findobj(mlaptr, 'Tag', 'Theta');
   set(fieldptr, 'UserData', default(1));
   fieldptr = findobj(mlaptr, 'Tag', 'Phi');
   set(fieldptr, 'UserData', default(2));
   uiresume(mlaptr);
end
%end MLACode


function num = StrCheck(str)
if ((str == 'i') | (str == 'j')) %Eliminate imaginary values.
   temp = [];
else
   temp = str2num(str);
end
```

```
if (isempty(temp) | (temp <= 0))
   errordlg('Felaktig inmatning!', 'Ett fel uppstod');
   errptr = findobj(0, 'Tag', 'Ett fel uppstod');
   set(errptr, 'WindowStyle', 'modal');
   num = 'err';
else
   num = temp;
end
%end StrCheck
```

MLECode.m

```
function MLECode(action)

set(0, 'ShowHiddenHandles', 'on');
startptr = findobj(0, 'Tag', 'StartWindow');
mleptr = findobj(0, 'Tag', 'MLEWindow');

switch(action)
case 'theta'
   fieldptr = findobj(mleptr, 'Tag', 'Phi');
   phi = get(fieldptr, 'UserData')*pi/180;
   str = get(gcbo, 'String');
   temp = StrCheck(action, str, phi);
   if (temp == 'err')
      default = get(gcbf, 'UserData'); %Input incorrect -> use
default value.
      set(gcbo, 'String', num2str(default(1))); %Update text
field.
      set(gcbo, 'UserData', default(1)); %Update working value.
      theta = default(1)*pi/180;
   else
      set(gcbo, 'UserData', temp); %Update working value.
      theta = temp*pi/180;
   end
case 'phi' %See comments for 'theta' above.
   fieldptr = findobj(mleptr, 'Tag', 'Theta');
   theta = get(fieldptr, 'UserData')*pi/180;
   str = get(gcbo, 'String');
   temp = StrCheck(action, str, theta);
   if (temp == 'err')
      default = get(gcbf, 'UserData');
      set(gcbo, 'String', num2str(default(2)));
      set(gcbo, 'UserData', default(2));
      phi = default(2)*pi/180;
   else
      set(gcbo, 'UserData', temp);
      phi = temp*pi/180;
   end
case 'ok'
   uiresume(mleptr);
case 'cancel'
   default = get(gcbf, 'UserData'); %Restore to default values.
   fieldptr = findobj(mleptr, 'Tag', 'Theta');
   set(fieldptr, 'UserData', default(1));
   fieldptr = findobj(mleptr, 'Tag', 'Phi');
   set(fieldptr, 'UserData', default(2));
```

```
   uiresume(mleptr);
end
%end MLECode


function num = StrCheck(action, str, angle)
if ((str == 'i') | (str == 'j')) %Eliminate imaginary values.
   temp = [];
else
   temp = str2num(str);
end

switch(action)
case 'theta'
   phi = angle*180/pi;
   if (isempty(temp) | (temp < 0) | (temp >= phi))
      errordlg('Felaktig inmatning!', 'Ett fel uppstod');
      errptr = findobj(0, 'Tag', 'Ett fel uppstod');
      set(errptr, 'WindowStyle', 'modal');
      num = 'err';
   else
      num = temp;
   end
case 'phi'
   theta = angle*180/pi;
   if (isempty(temp) | (temp < 0) | (temp <= theta))
      errordlg('Felaktig inmatning!', 'Ett fel uppstod');
      errptr = findobj(0, 'Tag', 'Ett fel uppstod');
      set(errptr, 'WindowStyle', 'modal');
      num = 'err';
   else
      num = temp;
   end
end
%end StrCheck
```

JammerPositions.m

```
function positions = JammerPositions()

PositionWindow;
hold on;
set(0, 'ShowHiddenHandles', 'on');
startptr = findobj(0, 'Tag', 'StartWindow');
objptr = findobj(startptr, 'Tag', 'Position');
posptr = findobj(0, 'Tag', 'PositionWindow');
positions = get(objptr, 'UserData');
set(posptr, 'UserData', positions);
set(posptr, 'WindowStyle', 'modal');
areaptr = findobj(posptr, 'Tag', 'PositionArea');
objptr = findobj(startptr, 'Tag', 'Height');
height = get(objptr, 'UserData');
objptr = findobj(startptr, 'Tag', 'Range');
range = get(objptr, 'UserData');
alpha = atan(range/height);
objptr = findobj(startptr, 'Tag', 'RadarMainLobe');
angles = get(objptr, 'UserData');
```

```
theta = angles(1)*pi/180;
choise = get(objptr, 'Value');
switch(choise)
case 1
   a = (height*tan(alpha+theta/2) - height*tan(alpha-
theta/2))/2;
   center = height*tan(alpha-theta/2) + a;
case 2
   phi = angles(2)*pi/180;
   a = height*(tan(phi) - tan(theta))/2;
   center = height*tan(theta) + a;
end

xmin = center-2*a; %Calculate and draw the swathwidth.
xmax = center+2*a;
ymin = -2*a/1.4;
ymax = 2*a/1.4;
xratio = (xmax - xmin)/532;
yratio = (ymax - ymin)/380;
axis([xmin, xmax, ymin, ymax]);
patchptr = patch([center-a, center+a, center+a, center-a],
[ymin, ymin, ymax, ymax], [0.9 0.9 0.9]);
set(patchptr, 'EdgeColor', 'none');
set(patchptr, 'ButtonDownFcn', 'PositionCode position');
plot([center-a, center-a], [ymin, ymax], 'k:', [center+a,
center+a], [ymin, ymax], 'k:');
plot([0, 0], [ymin, ymax], 'k-', [-4*xratio, 0], [0,
12*yratio], 'k-', [0, 4*xratio], [12*yratio, 0], 'k-');
text(0 - 10*xratio, 0, 'Flygriktning', 'HorizontalAlignment',
'center', 'Rotation', 90);

njammers = length(positions)/2; %Draw +'s representing the
jammers.
for i = 1:1:njammers
   x = positions(2*i - 1);
   y = positions(2*i);
   plot(x, y, 'k+');
end

uiwait(posptr);
if (ishandle(posptr))
   temp = get(posptr, 'UserData');
   close(posptr);
else
   temp = [];
end
positions = temp;
%end JammerPositions
```

PositionMovement.m

```
function PositionMovement()

%Calculate the cursor's position.
posptr = findobj(0, 'Tag', 'PositionWindow');
objptr = findobj(posptr, 'Tag', 'PositionArea');
winpos = get(posptr, 'Position');
axespos = get(objptr, 'Position');
```

```
xlim = get(objptr, 'Xlim');
ylim = get(objptr, 'Ylim');
xmin = winpos(1) + axespos(1);
ymin = winpos(2) + axespos(2);
xpixellength = axespos(3);
ypixellength = axespos(4);
xmetriclength = xlim(2) - xlim(1);
ymetriclength = ylim(2) - ylim(1);
xratio = xmetriclength/xpixellength;
yratio = ymetriclength/ypixellength;
pos = get(0, 'PointerLocation');
xpixelpos = pos(1) - xmin;
ypixelpos = pos(2)- ymin;
xmetricpos = round(xlim(1) + xratio*xpixelpos);
ymetricpos = round(ylim(1) + yratio*ypixelpos);

test = overobj('axes');
if (~isempty(test)) %Display the cursos's positoin.
   setptr(posptr, 'crosshair');
   objptr = findobj(posptr, 'Tag', 'Xpos');
   set(objptr, 'String', num2str(xmetricpos));
   objptr = findobj(posptr, 'Tag', 'Ypos');
   set(objptr, 'String', num2str(ymetricpos));
else
   setptr(posptr, 'arrow');
end
%end PositionMovement
```

PositionCode.m

```
function PositionCode(action)

set(0, 'ShowHiddenHandles', 'on');
posptr = findobj(0, 'Tag', 'PositionWindow');
objptr = findobj(posptr, 'Tag', 'PositionArea');
oldpos = get(posptr, 'UserData');
switch(action)
case 'position' %Store the current position.
   axis(axis);
   pos = get(objptr, 'CurrentPoint');
   x = pos(1,1);
   y = pos(1,2);
   newpos = [oldpos, round(x), round(y)];
   set(posptr, 'UserData', newpos);
   plot(x,y,'k+');
case 'ok'
   uiresume(posptr);
case 'clear'
   pos = get(posptr, 'UserData');
   lines = findobj(objptr, 'Type', 'line');
   njammers = length(lines) - 5;
   delete(lines(1:njammers));
   set(posptr, 'UserData', []);
end
%end PositionCode
```

JammerGain.m

```
function gain = JammerGain()
```

```
JammerGainWindow;
set(0, 'ShowHiddenHandles', 'on');
startptr = findobj(0, 'Tag', 'StartWindow');
gainptr = findobj(0, 'Tag', 'JammerGainWindow');
set(gainptr, 'WindowStyle', 'modal');
temp = get(startptr, 'UserData');
default = temp.jammergain;
set(gainptr, 'UserData', default);

objptr = findobj(gcbf, 'Tag', 'JammerGain');
temp = get(objptr, 'UserData');
for i=1:1:15 %Update working values.
   interval_i = ['Interval', num2str(i)];
   objptr = findobj(gainptr, 'Tag', interval_i);
   set(objptr, 'String', num2str(temp(i)));
   set(objptr, 'UserData', temp(i));
end

uiwait(gainptr);
if (ishandle(gainptr))
   temp = [];
   for i=1:1:15 %Collect input values.
      interval_i = ['Interval', num2str(i)];
      objptr = findobj(gainptr, 'Tag', interval_i);
      temp = [temp, get(objptr, 'UserData')];
   end
   close(gainptr);
else
   temp = default;
end

gain = temp;
%end JammerGain
```

JammerGainCode.m

```
function JammerGainCode(action)

set(0, 'ShowHiddenHandles', 'on');
gainptr = findobj(0, 'Tag', 'JammerGainWindow');
default = get(gainptr, 'UserData');

switch(action)
case 'interval1'
   str = get(gcbo, 'String');
   gain = GainCheck(str);
   if (gain == 'err')
      set(gcbo, 'String', num2str(default(1)));
      set(gcbo, 'UserData', default(1));
   else
      set(gcbo, 'UserData', gain);
   end
case 'interval2'
   str = get(gcbo, 'String');
   gain = GainCheck(str);
   if (gain == 'err')
      set(gcbo, 'String', num2str(default(2)));
```

```
        set(gcbo, 'UserData', default(2));
    else
        set(gcbo, 'UserData', gain);
    end
case 'interval3'
    str = get(gcbo, 'String');
    gain = GainCheck(str);
    if (gain == 'err')
        set(gcbo, 'String', num2str(default(3)));
        set(gcbo, 'UserData', default(3));
    else
        set(gcbo, 'UserData', gain);
    end
case 'interval4'
    str = get(gcbo, 'String');
    gain = GainCheck(str);
    if (gain == 'err')
        set(gcbo, 'String', num2str(default(4)));
        set(gcbo, 'UserData', default(4));
    else
        set(gcbo, 'UserData', gain);
    end
case 'interval5'
    str = get(gcbo, 'String');
    gain = GainCheck(str);
    if (gain == 'err')
        set(gcbo, 'String', num2str(default(5)));
        set(gcbo, 'UserData', default(5));
    else
        set(gcbo, 'UserData', gain);
    end
case 'interval6'
    str = get(gcbo, 'String');
    gain = GainCheck(str);
    if (gain == 'err')
        set(gcbo, 'String', num2str(default(6)));
        set(gcbo, 'UserData', default(6));
    else
        set(gcbo, 'UserData', gain);
    end
case 'interval7'
    str = get(gcbo, 'String');
    gain = GainCheck(str);
    if (gain == 'err')
        set(gcbo, 'String', num2str(default(7)));
        set(gcbo, 'UserData', default(7));
    else
        set(gcbo, 'UserData', gain);
    end
case 'interval8'
    str = get(gcbo, 'String');
    gain = GainCheck(str);
    if (gain == 'err')
        set(gcbo, 'String', num2str(default(8)));
        set(gcbo, 'UserData', default(8));
    else
        set(gcbo, 'UserData', gain);
    end
```

```
case 'interval9'
   str = get(gcbo, 'String');
   gain = GainCheck(str);
   if (gain == 'err')
      set(gcbo, 'String', num2str(default(9)));
      set(gcbo, 'UserData', default(9));
   else
      set(gcbo, 'UserData', gain);
   end
case 'interval10'
   str = get(gcbo, 'String');
   gain = GainCheck(str);
   if (gain == 'err')
      set(gcbo, 'String', num2str(default(10)));
      set(gcbo, 'UserData', default(10));
   else
      set(gcbo, 'UserData', gain);
   end
case 'interval11'
   str = get(gcbo, 'String');
   gain = GainCheck(str);
   if (gain == 'err')
      set(gcbo, 'String', num2str(default(11)));
      set(gcbo, 'UserData', default(11));
   else
      set(gcbo, 'UserData', gain);
   end
case 'interval12'
   str = get(gcbo, 'String');
   gain = GainCheck(str);
   if (gain == 'err')
      set(gcbo, 'String', num2str(default(12)));
      set(gcbo, 'UserData', default(12));
   else
      set(gcbo, 'UserData', gain);
   end
case 'interval13'
   str = get(gcbo, 'String');
   gain = GainCheck(str);
   if (gain == 'err')
      set(gcbo, 'String', num2str(default(13)));
      set(gcbo, 'UserData', default(13));
   else
      set(gcbo, 'UserData', gain);
   end
case 'interval14'
   str = get(gcbo, 'String');
   gain = GainCheck(str);
   if (gain == 'err')
      set(gcbo, 'String', num2str(default(14)));
      set(gcbo, 'UserData', default(14));
   else
      set(gcbo, 'UserData', gain);
   end
case 'interval15'
   str = get(gcbo, 'String');
   gain = GainCheck(str);
   if (gain == 'err')
```

```
      set(gcbo, 'String', num2str(default(15)));
      set(gcbo, 'UserData', default(15));
   else
      set(gcbo, 'UserData', gain);
   end
case 'ok'
   uiresume(gainptr);
case 'cancel'
   for i=1:1:15 %Restore to default values.
      interval_i = ['Interval', num2str(i)];
      objptr = findobj(gainptr, 'Tag', interval_i);
      set(objptr, 'UserData', default(i));
   end
   uiresume(gainptr);
end
%end JammerGainCode



function num = GainCheck(str)
if ((str == 'i') | (str == 'j')) %Eliminate imaginary values.
   temp = [];
else
   temp = str2num(str);
end

if (isempty(temp) | (temp > 25) | (temp < -25))
   errordlg('Felaktig inmatning!', 'Ett fel uppstod');
   errptr = findobj(0, 'Tag', 'Ett fel uppstod');
   set(errptr, 'WindowStyle', 'modal');
   num = 'err';
else
   num = temp;
end
%end GainCheck
```

ShowResult.m

```
function ShowResult(input)

height = input.height;
theta = input.radarlobe(2)*pi/180;
choice = input.radarlobe(1);
switch(choice)
case 1
   range = input.range;
   alpha = atan(range/height);
   a = height*(tan(alpha+theta/2) - tan(alpha-theta/2))/2;
   center = height*tan(alpha-theta/2) + a;
case 2
   phi = input.radarlobe(3)*pi/180;
   a = height*(tan(phi) - tan(theta))/2;
   center = height*tan(theta) + a;
end
radarpower = input.radarpower;
radargain = input.radargain;
jammerpower = input.jammerpower;
```

```
jamdepth = input.repetitions*input.pulseduration*300;
jamtime = input.jamtime/100;
groundgain = 10^-(input.groundgain/10);
bandratio = input.bandratio;

xmin = center-4*a;
xmax = center+4*a;
ymin = -4*a/1.4;
ymax = 4*a/1.4;
xratio = (xmax - xmin)/532;
yratio = (ymax - ymin)/380;

values = Initiation(input.jammerpositions, height, xmin, xmax,
ymin, ymax);
ystart = values.ystart;
ystop = values.ystop;
jammers = values.jammers;

njammers = length(jammers);
for y = ystart:yratio:ystop
   for i = 1:1:njammers
      xpos = jammers(i).xpos;
      ypos = jammers(i).ypos;
      beta = atan((y - ypos)/sqrt(height^2 + xpos^2))*180/pi;
      test = 0;
      for j = -45:3:42
         if ((beta >= j) & (beta <= j+3))
            if (j < 0)
               jammergain = 10^(input.jammergain(abs(j)/3)/10);
            else
               jammergain = 10^(input.jammergain(j/3 + 1)/10);
            end
            pixel.power =
(jammerpower*jammergain*jamtime*bandratio)/(4*pi*(xpos^2 + (y -
ypos)^2 + height^2));
            test = 1;
         end
      end

      if (test == 1) %Assign 'pixel.x' & 'pixel.y' only if
'pixel.gain' is assigned.
         pixel.x = sqrt(xpos^2 + (y-ypos)^2);
         pixel.y = y;
         pixel.jamdepth = jamdepth;
         jammers(i).pixels = [jammers(i).pixels, pixel];
      end
   end
end


for i = 1:1:njammers - 1
   ixpos = jammers(i).xpos;
   iypos = jammers(i).ypos;
   for j = i + 1:1:njammers
      pixels =[];
      jxpos = jammers(j).xpos;
      jypos = jammers(j).ypos;
      yupper = jypos + sqrt(height^2 + jxpos^2);
```

```
      ylower = jypos - sqrt(height^2 + jxpos^2);
      npixels = length(jammers(i).pixels);
      for k = 1:1:npixels
         pixel = [];
         x = jammers(i).pixels(k).x;
         y = jammers(i).pixels(k).y;
         if ((y >= ylower) & (y <= yupper))
               jx = sqrt(jxpos^2 + (y-jypos)^2);
               if ((x + jammers(i).pixels(k).jamdepth >= jx +
jamdepth) & (x <= jx + jamdepth))
                  pixel.power = jammers(i).pixels(k).power +
FindPower(y, jammers(j).pixels);
                  pixel.x = x;
                  pixel.y = y;
                  pixel.jamdepth = jx + jamdepth - x;
                  pixels = [pixels, pixel];

                  pixel.power = jammers(i).pixels(k).power;
                  pixel.x = jx + jamdepth;
                  pixel.y = y;
                  pixel.jamdepth = x +
jammers(i).pixels(k).jamdepth - jx - jamdepth;
                  pixels = [pixels, pixel];
               elseif ((x >= jx) & (x +
jammers(i).pixels(k).jamdepth <= jx + jamdepth))
                  jammers(i).pixels(k).power =
jammers(i).pixels(k).power + FindPower(y, jammers(j).pixels);
                  pixels = [pixels, jammers(i).pixels(k)];
               elseif ((x + jammers(i).pixels(k).jamdepth >=
jx) & (x <= jx))
                  pixel.power = jammers(i).pixels(k).power;
                  pixel.x = x;
                  pixel.y = y;
                  pixel.jamdepth = jx - x;
                  pixels = [pixels, pixel];

                  pixel.power = jammers(i).pixels(k).power +
FindPower(y, jammers(j).pixels);
                  pixel.x = jx;
                  pixel.y = y;
                  pixel.jamdepth = x +
jammers(i).pixels(k).jamdepth - jx;
                  pixels = [pixels, pixel];
               else
                  pixels = [pixels, jammers(i).pixels(k)];
               end
         else
            pixels = [pixels, jammers(i).pixels(k)];
         end
      end %for-loop with index 'k'.
      jammers(i).pixels = pixels;
   end %for-loop with index 'j'.
end %for-loop with index 'i'.
```

```
resptr = figure('Color', [0.7, 0.7, 0.7], 'Name',
'Störövervikt', 'MenuBar', 'none', 'Numbertitle', 'off',
'Position', [346 479 610 500], 'Units', 'pixels');
axesptr = axes('Units', 'pixels', 'Xlim', [xmin xmax], 'Ylim',
[ymin ymax]);
set(axesptr, 'Position', [56 73 532 380]);
hold on;
patchptr = patch([center-a, center+a, center+a, center-a],
[ymin, ymin, ymax, ymax], [0.9 0.9 0.9]);
set(patchptr, 'EdgeColor', 'none');
plot([center-a, center-a], [ymin, ymax], 'k:', [center+a,
center+a], [ymin, ymax], 'k:');
plot([0, 0], [ymin, ymax], 'k-', [-4*xratio, 0], [0,
12*yratio], 'k-', [0, 4*xratio], [12*yratio, 0], 'k-');
text(0 - 10*xratio, 0, 'Flygriktning', 'HorizontalAlignment',
'center', 'Rotation', 90);

extremepowers = FindExtremePowers(jammers);
minpower = extremepowers.minpower;
maxpower = extremepowers.maxpower;

dBmin = inf;
dBmax = -inf;
njammers = length(jammers);
for i = njammers:-1:1
   xpos = jammers(i).xpos;
   ypos = jammers(i).ypos;
   npixels = length(jammers(i).pixels);
   for j = 1:1:npixels
      x = jammers(i).pixels(j).x;
      y = jammers(i).pixels(j).y;
      targetdistance = xpos^2 + (y - ypos)^2 + height^2; %The
distance squared.
      receivedradarpower =
(radarpower*radargain*groundgain)/(4*pi*targetdistance);
      dBtemp =
10*log10(jammers(i).pixels(j).power/receivedradarpower);
      if (dBtemp < dBmin)
         dBmin = dBtemp;
      end
      if (dBtemp > dBmax)
         dBmax = dBtemp;
      end

      if (jammers(i).pixels(j).power > receivedradarpower)
%Compare the received effects.
         jamdepth = jammers(i).pixels(j).jamdepth;
         color = GetColor(maxpower, minpower,
jammers(i).pixels(j).power);
         line([x, x+jamdepth], [y+yratio, y+yratio], 'Color',
color, 'LineStyle', '-');
      end
   end
end

for i = 1:1:njammers
   xpos = jammers(i).xpos;
   ypos = jammers(i).ypos;
```

```
   yupper = ypos + sqrt(height^2 + xpos^2);
   ylower = ypos - sqrt(height^2 + xpos^2);
   if ((xmin <= xpos) & (xpos <= xmax) & (ymin <= ypos) & (ypos
<= ymax))
      plot(xpos, ypos, 'k+');
   end
end

helpptr = helpdlg(['Störövervikten ligger i intervallet [',
int2str(round(dBmin)), ', ', int2str(round(dBmax)), '] dB.'],
'dB-intervall');
%end ShowResult


function values = Initiation(positions, height, xmin, xmax,
ymin, ymax)
ystart = ymax;
ystop = ymin;
jammers =[];
njammers = length(positions)/2;
for i = 1:1:njammers
   xpos = positions(2*i - 1);
   ypos = positions(2*i);
   if ((xpos >= 0) & (xpos >= xmin) & (xpos <= xmax) & (ypos >=
ymin) & (ypos <= ymax))
      jammer.xpos = xpos;
      jammer.ypos = ypos;
      jammer.pixels = [];
      ylower = ypos - sqrt(height^2 + xpos^2);
      if (ylower < ystart)
         ystart = ylower;
      end
      yupper = ypos + sqrt(height^2 + xpos^2);
      if (yupper > ystop)
         ystop = yupper;
      end
   end
   jammers = [jammers, jammer];
end

if (ystart < ymin)
   ystart = ymin;
end
if (ystop > ymax)
   ystop = ymax;
end

values.ystart = ystart;
values.ystop = ystop;
values.jammers = jammers;
%end Initiation


function gain = FindPower(y, pixels)
for i = 1:1:length(pixels)
   if (y == pixels(i).y)
      temp = pixels(i).power;
```

```
   end
end

gain = temp;
%end FindPower



function extremepowers = FindExtremePowers(jammers)
extremepowers.maxpower = -inf;
extremepowers.minpower = inf;
for i = 1:1:length(jammers)
   for j = 1:1:length(jammers(i).pixels)
      if (jammers(i).pixels(j).power > extremepowers.maxpower)
         extremepowers.maxpower = jammers(i).pixels(j).power;
      end
      if (jammers(i).pixels(j).power < extremepowers.minpower)
         extremepowers.minpower = jammers(i).pixels(j).power;
      end
   end
end
%end FindExtremePowers



function color = GetColor(maxpower, minpower, power)
if (maxpower == minpower) %Avoid dividing by zero.
         cratio = 1;
else
   cratio = 1 - (power - minpower)/(maxpower - minpower);
end

if (cratio <= 0.25)
   red = 1;
   green = 4*cratio;
   blue = 0;
elseif ((cratio > 0.25) & (cratio <= 0.5))
   red = 2 - 4*cratio;
   green = 1;
   blue = 0;
elseif ((cratio > 0.5) & (cratio <= 0.75))
   red = 0;
   green = 1;
   blue = 4*cratio - 2;
else
   red = 0;
   green = 4 - 4*cratio;
   blue = 1;
end

color = [red, green, blue];
%end GetColor
```

Index