FFI-RAPPORT

17/16253

Product of two K-distributions

applications in automatic ship detection based on satellite SAR

—

Helge Knutsen

Product of two K-distributions applications in automatic ship detection based on satellite SAR

Helge Knutsen

Norwegian Defence Research Establishment (FFI) 18 December 2017

Keywords

Syntetisk apertur-radar (SAR)
Skipsdeteksjon
Statistikk
Polarisasjon
Havoverflaten

FFI-RAPPORT:
17/16253

Project number:
1441

ISBN:
P:978-82-464-3026-3
E:978-82-464-3027-0

Approved by:
Johnny Bardal, *Director*
Richard B. Olsen, *Research Manager*

Summary

In ship detection based on satellite Synthetic Aperture Radar (SAR) images, the intensity of the sea surface backscatter is modeled as a stochastic variable. For a single-polarization channel the K-distribution serves as a statistical model for the backscatter. Combining two channels by considering the product of the received intensities, it is desirable to determine a model for this new variable.

The main focus of this report is therefore to derive the probability distribution of the product of two independent K-distributed variables. By recognizing that a K-distributed variable is itself a product of two gamma-distributed variables, a four-product of gamma variables is considered instead. This distribution is determined by the means of the Mellin transform, which allows us to determine the distribution in general of an arbitrary product of such variables. Necessary background theory is presented before utilizing the transform. Although no explicit formula is derived, an implicit form is obtained and finally expressed in terms of the Meijer G-function.

The subsequent sections present a method to evaluate the distribution and produce look-up tables for the threshold values in the ship detection hypothesis test. Code and look-up tables are provided in Appendix A and B. Finally, possible caveats of the look-up tables are discussed, especially the sparseness of the parameter sets involved and the related accuracy of the threshold values. Some suggestions are made to improve upon the effective accuracy of the threshold values. Further investigation is required to improve the effective accuracy.

It also remains to investigate whether in fact combining two channels in a product is a more effective tool in ship detection rather than analyzing the signals from the channels separately.

Sammendrag

I skipsdeteksjon basert på SAR-bilder (syntetisk apertur-radar) er intensiteten til sjøoverflatestøy modellert som en stokastisk variabel. For en enkelt polarisasjonskanal fungerer K-fordelingen som en statistisk modell for sjøstøyen. Når to kanaler kombineres ved å vurdere produktet av mottatt intensitet, er det av interesse å utlede en modell for den nye sammenslåtte kanalen.

Hovedfokuset i denne rapporten er derfor å utlede sannsynlighetsfordelingen til produktet av to uavhengige K-fordelte variabler. Ved å erkjenne at en K-fordelt variabel selv er et produkt av to gamma-fordelte variabler, har vi derfor tatt tatt utgangspunkt i et fire-produkt av gamma-fordelte variabler i stedet. Denne fordelingen er bestemt ved hjelp av Mellin-tranformasjonen, som tillater oss å bestemme fordelingen til et vilkårlig produkt av slike variabler. Nødvendig bakgrunnsteori blir presentert før transformasjonen anvendes. Selv om ingen eksplisitt formel blir utledet, er fordelingen gitt på implisitt form, og til slutt uttrykt ved hjelp av Meijer G-funksjonen.

De påfølgende seksjonene presenterer en metode for å evaluere fordelingen og produsere tabellverk for terskelverdier i hypotesetesten ved skipsdeteksjon. Koden og tabellverket er tilgjengelig som vedlegg. Til slutt blir mulige utfordringer knyttet til tabellverket diskutert, spesielt spredningen av parameterverdiene relatert til presisjonen av beregnede terskelverdier. Enkelte forslag for å forbedre den effektive presisjonen til terskelverdiene blir presentert. Videre undersøkelser vil være nødvendig for å forbedre den effektive presisjonen.

Det gjenstår også å undersøke om det å kombinere to kanaler til ett produkt er et mer effektivt verktøy for skipsdeteksjon fremfor å analysere signalene fra kanalene separat.

Contents

1 Theory

Let Y be a K-distributed variable. The probability distribution is given by

$$p_Y(y) = \frac{2}{\Gamma(\beta_1)\Gamma(\beta_2)y}(\lambda_1\lambda_2 y)^{\frac{\beta_1+\beta_2}{2}} K_{\beta_1-\beta_2}(2\sqrt{\lambda_1\lambda_2 y}),\ y \geq 0 \tag{1.1}$$

where $\Gamma(\cdot)$ denotes the Gamma function and $K_n(\cdot)$ the modified Bessel function of the second kind of order n. The derivation of the above equation [1] is achieved by considering in return two Gamma-distributed variables, say X_1, X_2, each with probability distribution

$$p_{X_j}(x) = \frac{1}{\Gamma(\beta_j)}\lambda_j^{\beta_j} x^{\beta_j-1}\mathrm{e}^{-\lambda_j x},\ x \geq 0, \quad j = 1, 2. \tag{1.2}$$

Note that all parameters involved in both (1.1) and (1.2) are strictly positive. Thus, in order to determine the product-distribution of two K-distributed variables, we may instead consider the product of four Gamma-variables. This is ensured by the fact that regular multiplication is associative.

Hence, consider the general case where $X_1, X_2, ..., X_N$ are Gamma-distributed variables with distribution according to (1.2) with respective indexing j. The goal is to determine the distribution of $U = X_1 \cdot X_2 \cdot ... \cdot X_N$, for which we can later reduce to the special case $N = 4$.

1.1 Mellin transform in probability theory

Recall that the Fourier transform is a helpful tool to determine sums of stochastic variables. When dealing with products, however, we consider a different integral transform, namely the Mellin transform [2]. What proceeds are some preliminary definitions and results to provide sufficient background before applying the transform to our particular problem.

Def. 1.1.1. The Mellin norm of $f \in L_1(\mathbb{R}_+)$ is defined as

$$\|f\|_{\mathcal{M}_c} = \int_0^\infty |f(x)||x^{c-1}|dx, \text{ for some fixed } c \in \mathbb{C}. \tag{1.3}$$

Observe that in the above definition we did not restrict the norm to be less than infinity. Such a requirement leads to the following space.

Def. 1.1.2. The Mellin space $\mathcal{M}_c(\mathbb{R}_+)$ for some fixed $c \in \mathbb{C}$ consists precisely of the functions in $L_1(\mathbb{R}_+)$ for which the Mellin norm $\|\cdot\|_{\mathcal{M}_c}$ is well-defined, that is, less than infinity.

From Def.1.1.1 it should be evident that if a function f belongs $\mathcal{M}_c(\mathbb{R}_+)$, then f also belongs to $\mathcal{M}_{c+it}(\mathbb{R}_+)$ for all $t \in \mathbb{R}$. The space $\mathcal{M}_D(\mathbb{R}_+)$ for $D \subset \mathbb{C}$ is naturally extended from the point-wise definition above. It is on such an extension we define the Mellin transform.

Def. 1.1.3. The Mellin transform of a function $f \in \mathcal{M}_{[a,b]}(\mathbb{R}_+)$ is given by

$$\mathcal{M}(f)(s) = \int_0^\infty f(x)x^{s-1}dx, \text{ for } s \in \mathbb{C}. \tag{1.4}$$

The conditions in the above definition on f ensures that $\mathcal{M}(f)(s)$ is well-defined for $s \in \{\alpha + i\beta | \alpha \in [a,b], \beta \in \mathbb{R}\}$.

The next theorem provides us with a formula for the inverse Mellin transform and sufficient conditions for the inverse to exist.

Theorem 1.1.1. Suppose $f \in \mathcal{M}_{[a,b]}(\mathbb{R}_+)$ with Mellin transform $\mathcal{M}(f)(s)$. Provided f is continuously differentiable at point x, the inverse Mellin transform can be expressed

$$f(x) = \frac{1}{2\pi i}\int_{c-i\infty}^{c+i\infty} \mathcal{M}(f)(s)x^{-s}ds, \text{ for all } c \in [a,b]. \tag{1.5}$$

Proof. Recall the Fourier transform of a function, say g, is given by

$$\hat{g}(\omega) = \int_{-\infty}^{\infty} g(x)\mathrm{e}^{-i\omega x}dx,$$

and the inverse transform

$$\check{g}(x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} g(x)\mathrm{e}^{i\omega x}d\omega.$$

Take $c \in [a,b]$ and consider the operator T on g defined by $Tg(x) = g(\mathrm{e}^x)\mathrm{e}^{cx}$. It is easy to show that the inverse operator is given by $T^{-1}g(x) = g(\ln(x))x^{-c}$. We proceed by applying the Fourier transform to Tf, that is

$$\begin{aligned}
\widehat{Tf}(\omega) &= \int_{-\infty}^{\infty} Tf(x)\mathrm{e}^{-i\omega x}dx \\
&= \int_{-\infty}^{\infty} f(\mathrm{e}^x)\mathrm{e}^{x(c-i\omega)}dx \text{ with substitution } u = \mathrm{e}^x \\
&= \int_0^{\infty} f(u)u^{c-i\omega-1}du \text{ which we recognize from Def.1.1.3 as} \\
&= \mathcal{M}(f)(c-i\omega).
\end{aligned}$$

By assumption, we have that $\mathcal{M}(f)(c-i\omega)$ is well-defined. Observe that since f is continuously differentiable, the same is true for Tf. Hence, by the Fourier inversion theorem [3] (Theorem 2.1), we get

$$\begin{aligned}
Tf(x) &= \frac{1}{2\pi}\int_{-\infty}^{\infty} \widehat{Tf}(\omega)\mathrm{e}^{i\omega x}d\omega \\
&= \frac{1}{2\pi}\int_{-\infty}^{\infty} \mathcal{M}(f)(c-i\omega)\mathrm{e}^{i\omega x}d\omega \quad \text{with substitution } s = c - i\omega \\
&= \frac{1}{2\pi i}\int_{c-i\infty}^{c+i\infty} \mathcal{M}(f)(s)\mathrm{e}^{x(c-s)}ds.
\end{aligned}$$

Finally, apply the inverse operator T^{-1} to Tf such that

$$\begin{aligned}
f(x) &= T^{-1}(Tf)(x) \\
&= T^{-1}\frac{1}{2\pi i}\int_{c-i\infty}^{c+i\infty} \mathcal{M}(f)(s)\mathrm{e}^{x(c-s)}ds \\
&= \frac{1}{2\pi i}\int_{c-i\infty}^{c+i\infty} \mathcal{M}(f)(s)x^{c-s}x^{-c}ds \\
&= \frac{1}{2\pi i}\int_{c-i\infty}^{c+i\infty} \mathcal{M}(f)(s)x^{-s}ds.
\end{aligned}$$

Since $c \in [a,b]$ was arbitrary, the same result holds for all $c \in [a,b]$. □

We will now begin to relate these notions to probability theory, in particular to products of stochastic variables.

Consider two independent, continuous stochastic variables $X_1, X_2 \geq 0$ with probability distribution p_{X_1}, p_{X_2}, respectively. The product of these two variables, say $U = X_1 \cdot X_2$, has distribution determined by the integral

$$p_U(u) = \int_0^{\infty} p_{X_1}\left(\frac{u}{x}\right) p_{X_2}(x)\frac{dx}{x}. \tag{1.6}$$

This motivates the following definition.

Def. 1.1.4. For functions $f, g \in L_1(\mathbb{R})$ their Mellin convolution is given by

$$(f \diamond g)(u) = \int_0^{\infty} f\left(\frac{u}{x}\right) g(x)\frac{dx}{x}. \tag{1.7}$$

With this new definition available, $p_U(u)$ can be expressed as a Mellin convolution

$$p_U(u) = (p_{X_1} \diamond p_{X_2})(u). \tag{1.8}$$

Since regular multiplication of variables is both associative and commutative, we would naturally expect that the same should hold for the Mellin convolution. These two properties are summarized in the next lemma.

Lemma 1.1.2. The Mellin convolution is associative and commutative, that is for all $f, g, h \in L_1(\mathbb{R}_+)$

$$(f \diamond g) \diamond h = f \diamond (g \diamond h), \tag{1.9}$$
$$f \diamond g = g \diamond f. \tag{1.10}$$

Proof. Commutativity is verified directly by a simple substitution. Associativity is somewhat more involved

$$\begin{aligned}
((f \diamond g) \diamond h)(u) &= \int_0^\infty (f \diamond g)\left(\frac{u}{x}\right) h(x) \frac{dx}{x} \\
&= \int_0^\infty \left[\int_0^\infty f\left(\frac{u}{yx}\right) g(y) \frac{dy}{y}\right] h(x) \frac{dx}{x} \quad \text{with substitution } z = yx \\
&= \int_0^\infty \left[\int_0^\infty f\left(\frac{u}{z}\right) g\left(\frac{z}{x}\right) \frac{dz}{z}\right] h(x) \frac{dx}{x} \quad \text{by the Fubini} - \text{Tonelli theorem[4]} \\
&= \int_0^\infty f\left(\frac{u}{z}\right) \left[\int_0^\infty g\left(\frac{z}{x}\right) h(x) \frac{dx}{x}\right] \frac{dz}{z} \\
&= \int_0^\infty f\left(\frac{u}{z}\right) (g \diamond h)(z) \frac{dz}{z} \\
&= (f \diamond (g \diamond h))(u).
\end{aligned}$$

□

Similar to the regular convolution, we have a convolution theorem, but now in terms of the Mellin transform as opposed to the Fourier transform.

Theorem 1.1.3. Let $f, g \in \mathcal{M}_s(\mathbb{R})$, then

$$\mathcal{M}(f \diamond g)(s) = \mathcal{M}(f)(s) \cdot \mathcal{M}(g)(s). \tag{1.11}$$

Proof.

$$
\begin{aligned}
\mathcal{M}(f \diamond g)(s) &= \int_0^\infty (f \diamond g)(u) u^{s-1} du \\
&= \int_0^\infty \left[\int_0^\infty f\left(\frac{u}{x}\right) g(x) \frac{dx}{x} \right] u^{s-1} du \text{ by the Fubini – Tonelli theorem[4]} \\
&= \int_0^\infty \left[\int_0^\infty f\left(\frac{u}{x}\right) u^{s-1} du \right] g(x) \frac{dx}{x} \text{ with substitution } y = \frac{u}{x} \\
&= \int_0^\infty \left[\int_0^\infty f(y) y^{s-1} dy \right] g(x) x^{s-1} dx \\
&= \int_0^\infty \mathcal{M}(f)(s) g(x) x^{s-1} dx \\
&= \mathcal{M}(f)(s) \int_0^\infty g(x) x^{s-1} dx \\
&= \mathcal{M}(f)(s) \cdot \mathcal{M}(g)(s).
\end{aligned}
$$

□

This may easily be extended to an arbitrary, finite number of functions.

Corollary 1.1.3.1. Let $f_1, f_2, ..., f_N \in \mathcal{M}_s(\mathrm{R})$, then

$$\mathcal{M}(f_1 \diamond f_2 \diamond ... \diamond f_N)(s) = \mathcal{M}(f_1)(s) \cdot \mathcal{M}(f_2)(s) \cdot ... \cdot \mathcal{M}(f_N)(s). \tag{1.12}$$

Proof. From Lemma 1.1.2 we have that the Mellin convolution is associative. Thus, by induction on $\mathcal{M}((f_1 \diamond f_2 \diamond ... \diamond f_{N-1}) \diamond f_N)$ the desired result is obtained. □

It is this final Corollary which demonstrates the utility of the Mellin tranform with regard to products of stochastic variables. If we assume the variables involved are independent, continuous and positive, then their product distribution can be determined by considering the inverse Mellin transform of the regular product of Mellin transforms of the individual probability distributions. Hence, the remaining challenges consist of calculating the Mellin transform of the distributions and, perhaps more complicated, estimating the inverse of the product.

1.2 Product distribution of N Gamma-variables

As emphasized earlier, we will restrict our attention to products of Gamma-distributed variables $X_1, X_2, ..., X_N$. From (1.2) it is evident that $p_{X_j} \in \mathcal{M}_{[1,\infty[}$ for $j = 1, 2, ..., N$. We start by deriving the Mellin tranform of the single Gamma-distribution.

Lemma 1.2.1. Let X be a stochastic variable with Gamma distribution p_X according to (1.2), without the indexing. Then the Mellin transform of p_X is given by

$$\mathcal{M}(p_X)(s) = \frac{\lambda^{1-s}}{\Gamma(\beta)}\Gamma(\beta + s - 1). \tag{1.13}$$

Proof.

$$\begin{aligned}
\mathcal{M}(p_X)(s) &= \int_0^\infty p_X(x)x^{s-1}dx \\
&= \frac{\lambda^\beta}{\Gamma(\beta)}\int_0^\infty x^{\beta-1}\mathrm{e}^{-\lambda x}x^{s-1}dx \\
&= \frac{\lambda^\beta}{\Gamma(\beta)}\int_0^\infty x^{\beta+s-2}\mathrm{e}^{-\lambda x}dx \quad \text{with substitution } y = \lambda x \\
&= \frac{\lambda^{1-s}}{\Gamma(\beta)}\int_0^\infty y^{\beta+s-2}\mathrm{e}^{-y}dy \quad \text{where we recognize the Gamma function} \\
&= \frac{\lambda^{1-s}}{\Gamma(\beta)}\Gamma(\beta + s - 1).
\end{aligned}$$

□

With this lemma established, we are ready to formulate the main result.

Theorem 1.2.2. Let $X_1, X_2, ..., X_N$ be independent stochastic variables with Gamma-distribution p_{X_j} according to (1.2), with respective indexing j. Then the product $U = X_1 \cdot X_2 \cdot ... \cdot X_N$ has probability distribution

$$p_U(u) = \frac{1}{u}\prod_{j=1}^N \frac{1}{\Gamma(\beta_j)}\frac{1}{2\pi i}\int_{-i\infty}^{i\infty}\prod_{j=1}^N \Gamma(\beta_j + s)\left(u\prod_{j=1}^N \lambda_j\right)^{-s} ds, \; u \geq 0. \tag{1.14}$$

In particular for $N = 4$ we obtain the product distribution of two K-distributions.

Proof. From Corollary 1.1.3.1 we have

$$\begin{aligned}
\mathcal{M}(p_U)(s) &= \prod_{j=1}^N \mathcal{M}(p_{X_j})(s) \quad \text{which by Lemma 1.2.1} \\
&= \prod_{j=1}^N \frac{\lambda_j^{1-s}}{\Gamma(\beta_j)}\Gamma(\beta_j + s - 1).
\end{aligned}$$

Since $p_{X_j} \in \mathcal{M}_{[1,\infty[}$ for $j = 1, 2, ..., N$, the same holds true for p_U. Thus, from Theorem 1.1.1 the inverse Mellin transform of $\mathcal{M}(p_U)$ reads

$$\begin{aligned} p_U(u) = \mathcal{M}^{-1}\mathcal{M}(p_U)(u) &= \frac{1}{2\pi i}\int_{1-i\infty}^{1+i\infty} \mathcal{M}(p_U)(t)u^{-t}\,dt \\ &= \frac{1}{2\pi i}\int_{1-i\infty}^{1+i\infty} \prod_{j=1}^{N} \frac{\lambda_j^{1-t}}{\Gamma(\beta_j)}\Gamma(\beta_j + t - 1)u^{-t}\,dt \quad \text{with substitution } s = t - 1 \\ &= \frac{1}{u}\prod_{j=1}^{N}\frac{1}{\Gamma(\beta_j)}\frac{1}{2\pi i}\int_{-i\infty}^{i\infty}\prod_{j=1}^{N}\Gamma(\beta_j + s)\left(u\prod_{j=1}^{N}\lambda_j\right)^{-s} ds. \end{aligned}$$

□

Hence, from (1.14) we have an exact expression for the product distribution although on implicit form. In order to evaluate the probability distribution the integral in (1.14) needs to be estimated, either numerically or analytically. As is turns out, this integral belongs to a family of functions, namely the Meijer G-functions [5].

Def. 1.2.1. Let $m, n, p, q \in \mathbb{N} \cup \{0\}$ such that $m \leq q$ and $n \leq p$. Consider the set of scalars $a_1, ..., a_p, b_1, ..., b_q$ in $\mathbb{C}$ such that $a_k - b_j \notin \mathbb{N}$ for $k = 1, ..., n$ and $j = 1, ..., m$. Then the Meijer G-function at point $z \neq 0$ is given by

$$G_{p\,q}^{m\,n}\left({}^{a_1,...,a_p}_{b_1,...,b_q} \middle| z \right) = \frac{1}{2\pi i}\int_L \frac{\prod_{j=1}^{m}\Gamma(b_j + s)\prod_{j=1}^{n}\Gamma(1 - a_j - s)}{\prod_{j=m+1}^{q}\Gamma(1 - b_j - s)\prod_{j=n+1}^{p}\Gamma(a_j + s)} z^{-s}\,ds, \tag{1.15}$$

where L is one of three main paths in the complex plane. The path is such that the poles of $\prod_{j=1}^{m}\Gamma(b_j + s)$ are separated from the poles of $\prod_{j=1}^{m}\Gamma(1 - a_j - s)$. The Gamma function has poles where its argument equals a negative integer. Hence, the path separates the points

$$\{-b_j - r;\; r = 0, 1, 2, ... \text{ and } j = 1, ..., m\} \tag{1.16}$$

from

$$\{1 - a_j + r;\; r = 0, 1, 2, ... \text{ and } j = 1, ..., n\}. \tag{1.17}$$

One possible path L runs from $-i\infty$ to $i\infty$ such that (1.16) is kept to the left of L and consequently (1.17) to the right. Then the integral converges absolutely when

$$|\arg(z)| < (m + n - \frac{1}{2}(p + q)). \tag{1.18}$$

Observe that any empty product is by convention unity, that is, given numerical value 1.

By inspection of (1.14), we observe that the product distribution p_U may be expressed compactly in terms of the Meijer G-function

$$p_U(u) = \frac{1}{u}\prod_{j=1}^{N}\frac{1}{\Gamma(\beta_j)} G^{N\ 0}_{0\ N}\left(\begin{matrix} - \\ \beta_1,\ldots,\beta_N \end{matrix}\middle| u\prod_{j=1}^{N}\lambda_j\right),\ u \geq 0. \tag{1.19}$$

Hence, for the product of two K-distributions p_U reads

$$p_U(u) = \frac{1}{u}\prod_{j=1}^{4}\frac{1}{\Gamma(\beta_j)} G^{4\ 0}_{0\ 4}\left(\begin{matrix} - \\ \beta_1,\ldots,\beta_4 \end{matrix}\middle| u\prod_{j=1}^{4}\lambda_j\right),\ u \geq 0. \tag{1.20}$$

For $u = 0$ we take the limiting value of p_U as $u \to 0^+$.

Although this classification may seem redundant, recognizing the Meijer G-function in the probability distribution yields advantages due to the developed theory for this family of functions.

1.3 Cumulative distribution of N Gamma-variables

In applications, the cumulative distribution of a probability distribution is of particular importance. Recall, that the cumulative distribution of a stochastic variable $U \geq 0$ with probability distribution $p_U(x)$ evaluated at point u is given

$$P_U(u) = \int_0^u p_U(x)dx, \tag{1.21}$$

that is the probability that the variable takes values below threshold u. In order to derive the cumulative distribution of U with distribution according to (1.20), we make use of the following identity.

Lemma 1.3.1. Let $\rho, \sigma \in \mathbb{C}$ such that $\mathrm{Re}(\sigma) > 0$ and $\mathrm{Re}(\rho + b_j) > 0$, for $j = 1, \ldots, m$ according to Def. 1.2.1. Then

$$\int_0^1 x^{\rho-1}(1-x)^{\sigma-1} G^{m\ n}_{p\ q}\left(\begin{matrix} a_1,\ldots,a_p \\ b_1,\ldots,b_q \end{matrix}\middle| \omega x\right) dx = \Gamma(\sigma) G^{m\ \ n+1}_{p+1\ q+1}\left(\begin{matrix} 1-\rho, a_1,\ldots,a_p \\ b_1,\ldots,b_q, 1-\rho-\sigma \end{matrix}\middle| \omega\right), \tag{1.22}$$

under the same convergence condition as in (1.18).

Proof. See Saxena [5] Chapter 3 p.98. □

Theorem 1.3.2. Let $X_1, X_2, \ldots, X_N$ be independent stochastic variables with Gamma distribution p_{X_j} according to (1.2), with respective indexing j. Then the product $U = X_1 \cdot X_2 \cdot \ldots \cdot X_N$ has cumulative distribution

$$P_U(u) = \prod_{j=1}^{N}\frac{1}{\Gamma(\beta_j)} G^{N\ \ 1}_{1\ N+1}\left(\begin{matrix} 1, - \\ \beta_1,\ldots,\beta_q, 0 \end{matrix}\middle| u\prod_{j=1}^{N}\lambda_j\right),\ u \geq 0. \tag{1.23}$$

Proof. From the definition of cumulative distribution we have

$$
\begin{aligned}
P_U(u) &= \int_0^u p_U(x)dx, \text{ which when inserting (1.20)} \\
&= \prod_{j=1}^{N} \frac{1}{\Gamma(\beta_j)} \int_0^u \frac{1}{x} G^{N\ 0}_{0\ N}\left({- \atop \beta_1,\ldots,\beta_N} \middle| x \prod_{j=1}^{N} \lambda_j \right) dx \text{ with substitution } x = ut \\
&= \prod_{j=1}^{N} \frac{1}{\Gamma(\beta_j)} \int_0^1 t^{-1} G^{N\ 0}_{0\ N}\left({- \atop \beta_1,\ldots,\beta_N} \middle| ut \prod_{j=1}^{N} \lambda_j \right) dt.
\end{aligned}
$$

By comparison we recognize the integral in the above equation equals the left-hand-side of (1.22) with $\rho = 0$, $\sigma = 1$ and $\omega = u \prod_{j=1}^{N} \lambda_j$. Since $\beta_j > 0$ by assumption, we may apply Lemma 1.3.1, for which we attain the desired result. □

From this latest theorem we deduce the cumulative distribution for the product of two K-distributions

$$
P_U(u) = \prod_{j=1}^{4} \frac{1}{\Gamma(\beta_j)} G^{4\ 1}_{1\ 5}\left({1, - \atop \beta_1,\ldots,\beta_4, 0} \middle| u \prod_{j=1}^{4} \lambda_j \right), \ u \geq 0. \tag{1.24}
$$

2 Application

In Brekke's report [6] (equation (3.2)) the K-distribution is presented as a statistical model for the sea surface backscatter for a single channel. Compared with (1.1) the parameters are expressed somewhat differently with a more physical interpretation. In summary β_1, β_2 are replaced with

- L the Equivalent Number of Looks (ENL)
- ν the order parameter.

Also $\lambda_1 \cdot \lambda_2$ is replaced with $\frac{L\nu}{\mu}$, where μ is the mean of the distribution. Furthermore, numerical schemes are presented in order to estimate L, ν, μ from the image data.

Combining two channels, however, yields a probability and cumulative distribution for the backscatter according to (1.20) and (1.24), respectively. In this product distribution the same parameters are involved, with modifications as described above, but now with additional indexing $j = 1, 2$ to emphasize the channel. Hence, the probability distribution of the product becomes

$$p_U(u) = \frac{1}{u}\frac{1}{\Gamma(L_1)\Gamma(L_2)\Gamma(\nu_1)\Gamma(\nu_2)} G^{4\,0}_{0\,4}\left({- \atop L_1, L_2, \nu_1, \nu_2} \middle| u \cdot \frac{L_1 L_2 \nu_1 \nu_2}{\mu_1 \mu_2} \right) \tag{2.1}$$

and the cumulative distribution

$$P_U(u) = \frac{1}{\Gamma(L_1)\Gamma(L_2)\Gamma(\nu_1)\Gamma(\nu_2)} G^{4\,1}_{1\,5}\left({1, - \atop L_1, L_2, \nu_1, \nu_2} \middle| u \cdot \frac{L_1 L_2 \nu_1 \nu_2}{\mu_1 \mu_2} \right). \tag{2.2}$$

The same numerical schemes as for the K-distribution may be performed, on the individual images, to determine the parameters in the new distribution.

2.1 Hypothesis test in ship detection

Similar to Brekke's report, a type 1 hypothesis test can be carried out to distinguish signal from a vessel versus backscatter. Suppose the received intensity signal u belongs to the backscatter. Then the probability of an intensity larger or equal than measured u is

$$1 - P_U(u). \tag{2.3}$$

Thus, for sufficiently large u, such a signal becomes increasingly improbable and is expected to belong to a vessel rather than backscatter. Fix the constant false alarm rate (CFAR), that is the probability the signal belongs to sea surface backscatter. For a given CFAR there is a corresponding intensity value t according to

$$\text{CFAR} = 1 - P_U(t). \tag{2.4}$$

Any received signal above this threshold t is labeled as a detected vessel. Observe that while CFAR is user specified, the threshold is not and depends on the parameters in the distribution. The goal is therefore to determine the threshold, which is achieved by solving (2.4) for t.

For fast processing, it is desirable to create look-up tables of threshold values with different parameters L_1, L_2, ν_1, ν_2. The estimated parameters may then be approximated to the nearest value in the set of parameters in the tables. Observe that the threshold values should be normalized, that is the mean $\mu_1 \cdot \mu_2$ is set to 1. For distributions with mean $\mu_1 \cdot \mu_2 \neq 1$, the non-normalized threshold value T is obtained from the normalized t by

$$T = t \cdot \mu_1 \mu_2. \tag{2.5}$$

2.2 Implementation of threshold values

The main challenge of determining the threshold relates to our ability to evaluate the Meijer G-function in (2.2). If possible, we should take advantage of available software and already built-in tools related to the Meijer G-function. In python, Fredrik Johansson et al have developed a library, mpmath, with a built-in Meijer G-function, meijerg() [7]. This library has been utilized in the subsequent calculations. The necessary python-code is provided in Appendix A.1, and the main parts will be outlined in what follows.

Once able to evaluate the Meijer G-function, a numerical scheme is applied to (2.4) to approximate the threshold. A simple binary search for the threshold has been implemented in the function search_threshold(). The convergence of the method is guaranteed by the fact that the cumulative distribution is continuous. For an initial guess I_0 and uncertainty δ, the binary search has running time $\mathcal{O}(\log(\frac{I_0}{\delta}))$. However, the function only searches for the threshold in the interval $]0, 2I_0/(L_1 L_2 \nu_1 \nu_2)[$. Therefore the initial guess should be sufficiently large such that the interval contains the true solution.

The function multi_threshold() calls search_threshold() and searches for threshold values for all combinations of parameter values in user specified discrete sets (line 119, 120). These thresholds and parameter sets are later then saved to separate text-files in save_threshold(). Notice that it is assumed that the order parameter and the equivalent number of looks take upon values from separate sets. Still, the order parameter for the two channels are presumed to take values from the same set, similarly for the equivalent number of looks.

Now, suppose there are m choices for the order parameter and n choices for the equivalent number of looks. This yields a total of $(m \cdot n)^2$ combinations of the parameters. However, not all of the combinations return unique threshold values. Due to symmetry of (2.2) with respect to the parameters, the threshold is invariant to any permutation of the parameters. In particular it does not matter whether $\nu_1 = a, \nu_2 = b$ or $\nu_1 = b, \nu_2 = a$. This symmetry has been exploited in the code, with additional indexing, to reduce the running time.

The function check() was originally intended to verify that the symmetry indexing is performed correctly. Once verified under implementation, it now serves as an initial quality check for the searches. If for a single estimated threshold t the corresponding quantile $1 - P_U(t)$ is not within the interval $]0.99, 1.01[$CFAR, the function returns -1. This implies that either the initial guess or the accuracy is too low. If the initial guess is in fact properly established, then the accuracy must be increased, achieved by reducing delta (line 122).

Preliminary tests show that the initial guess must be increased if the parameters in the distribution increase. If true, it is then sufficient to establish the initial guess for the largest parameter values in the user specified sets. Observe, users are advised to perform some tests on suitable initial guess in advance to avoid unnecessary large initial guess, e.g. 100 fold greater than largest threshold value scaled with factor $L_1 L_2 \nu_1 \nu_2$. Large initial guess will increase the running time for both the binary search and the Meijer G-evaluation in meijerg().

3 Results and discussion

In Appendix B.1, B.2 two look-up tables are presented for the threshold values of the product distribution with CFAR = 10^{-7}, 10^{-8} respectively[1]. The calculated threshold values have an uncertainty of $\pm 10^{-8}$. The choice for the parameter sets are based on the parameter sets in Brekke's report [6], that is $\mathrm{ENL}_j = 1.0,\ 2.0,\ 3.0,\ 4.0$ and $\nu_j = 5.0,\ 10.0,\ 15.0,\ 20.0,\ 40.0,\ 90.0$ for $j = 1, 2$.

As an additional quality check of the algorithm, the look-up tables in Brekke's report in Appendix A has been recreated, located in Appendix B.3, B.4. The python-code is provided in Appendix A.2 and is structured similarly to the code for the product distribution. Brekke's table is somewhat incomplete, with some threshold values omitted and some require an update (A.2, block: ENL = 1.0). These values are nevertheless calculated in the new and updated look-up table. The re-estimated values carry an uncertainty of $\pm 10^{-8}$.

However, deviations from Brekke's table start as early as the sixth cipher. Although Brekke's estimated values are presented with 16 decimals, it is unclear whether the values in fact are estimated to such an accuracy. Hence from a conservative point of view, it is reasonable to conclude that the new implemented algorithm has at least five cipher accuracy. On the other hand, notice the current sparseness of the table parameters, e.g. $\nu = 5.0,\ 10.0,\ 15.0, \ldots$ instead of say $\nu = 5.0,\ 5.1,\ 5.2, \ldots$ Including the round-off of the parameters, the sparseness reflects an effective accuracy of about two ciphers for the threshold values. Hence, even a five cipher accuracy should seem sufficient for further use.

Returning to the look-up tables for the product distribution, the same effective accuracy problem related to parameter sparseness occurs here. A basic solution is simply to increase the density of the parameters in the user parameter sets. In practice, this might not be a tangible solution as an increase in possible parameters rapidly increases both the memory consumption and running time. Therefore, any extra parameter values should be added to capture large changes in the threshold value.

In figure 3.1 and figure 3.2 some of the threshold values are plotted as a function of the parameters ν_1 and L_1, respectively. The line between the points represents a linear approximation of the threshold between the discrete points. The threshold values will develop similarly for the $L_2,\ \nu_2$ or if $L_1,\ \nu_1$ are increased in the current plots. From these figures, it seems evident that the threshold decreases as the parameters increase. In addition the threshold function appears to have a positive curvature, that is the rate of decrease is decreasing. Suppose that the threshold function is smooth up to the second derivative for each parameter. Then a linear approximation between discrete points will lie above the actual curve. Thus, a linear approximation will systematically provide a larger threshold estimate. Furthermore, this approach will also improve the final estimated thresholds values compared to only nearest round-off.

This linear spline method yields a continuous approximation to the threshold function. However, the aforementioned approximation is not smooth. In order to capture possible smoothness of the function, higher order spline methods can be considered, say the cubic spline [8].

[1] Since the distribution is symmetric, as explained in 2.2, the symmetric values are not listed. In particular for $a > b$ where $\nu_1 = a,\ \nu_2 = b$, the corresponding threshold value will be listed at $\nu_1 = b,\ \nu_2 = a$, similarly for $\mathrm{ENL}_1,\ \mathrm{ENL}_2$

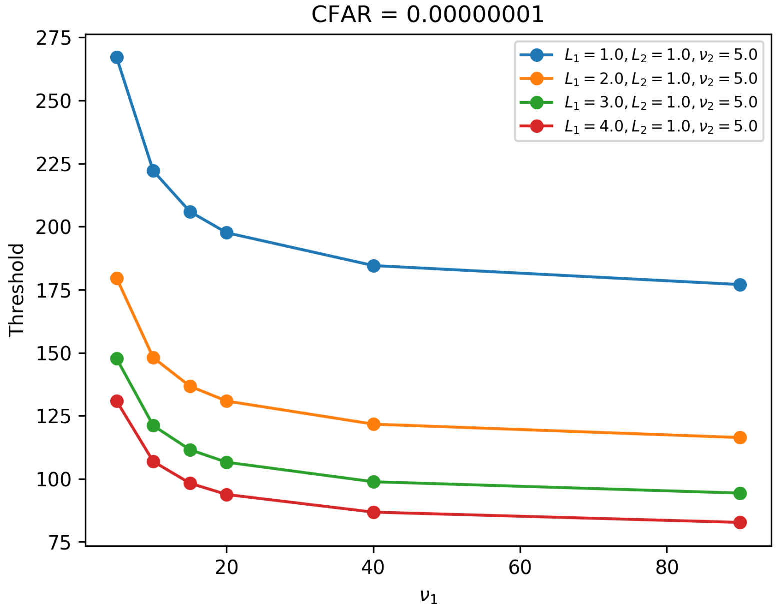


Figure 3.1 *Threshold values for* CFAR $= 10^{-8}$ *as a function of order parameter* ν_1 *for channel 1, where* $L_2 = 1.0$ *and* $\nu_2 = 5.0$.

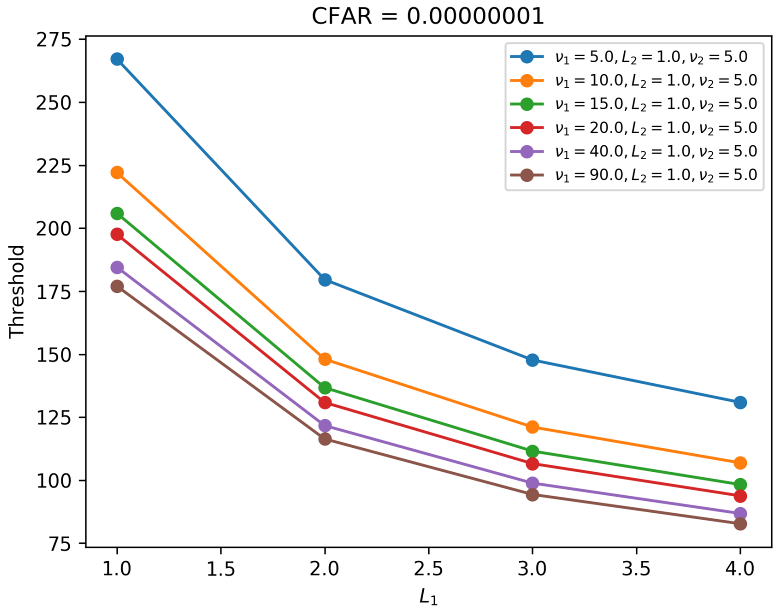


Figure 3.2 *Threshold values for* CFAR $= 10^{-8}$ *as a function of efficient number of looks* L_1 *for channel 1, where* $L_2 = 1.0$ *and* $\nu_2 = 5.0$.

4 Notation

The notation used in this text in pretty standard. However, I will list the main possible points of confusion

- The set of natural numbers $\mathbb{N}$ contains precisely the strictly positive integers, that is, zero is not included.
- The imaginary unit $\sqrt{-1}$ is denoted by i.
- The argument of a complex number z is denoted by $\arg(z)$.
- The real part of a complex number z is denoted by $\mathrm{Re}(z)$.
- The open interval from a to b in $\mathbb{R}$ is denoted by $]a, b[$ and the closed by $[a, b]$. Half-open-half-closed intervals are naturally extended from this.
- The $L_1(M)$ space of measure space M is the space of all absolutely integrable functions $f : M \to \mathbb{R}$ (or $\mathbb{C}$), that is, f is measurable and $\int_M |f(x)|dx < \infty$.
- For linguistic simplicity the probability density function of a continuous stochastic variable is referred to as the probability distribution or simply distribution. This should not be confused with the cumulative distribution which is always referred to as such.

Bibliography

[1] J.S. Lee, E. Pottier: Polarimetric Radar Imaging, From basics to applications. CRC Press, 2009 (Chapter 4.3.1)

[2] D. Collins: The relationship between Fourier and Mellin transforms, with applications to probability. University of New Mexico, Course: Wavelets 2007
http://www.math.unm.edu/ crisp/courses/wavelets/fall07/Mellin.pdf

[3] A. Boggess, F.J. Narcowich: A First Course in Wavelets with Fourier Analysis - 2nd ed. John Wiley & Sons, 2009 (Chapter 2.1.1)

[4] T. Tao: An Introduction to Measure Theory. Graduate Studies in Mathematics - Volume 126. American Mathematical Society, 2011 (Chapter 1.7, Corollary 1.7.23)

[5] A.M. Mathai, R.K. Saxena: Generalized Hypergeometric Functions with Applications in Statistics and Physical Sciences. Springer-Verlag Berlin·Heidelberg·New York, 1973 (Chapter 1.1)

[6] C. Brekke: Automatic ship detection based on satellite SAR. Forsvarets forskningsinstitutt/Norwergian Defence Research Establishment (FFI), 2009, FFI/Rapport-2008/00847

[7] F. Johansson et al: Python library mpmath, 2014
https://pypi.python.org/pypi/mpmath (download)
http://mpmath.org/doc/0.19/functions/hypergeometric.html (see Meijer G)

[8] E. W. Weisstein: Cubic Spline. From MathWorld–A Wolfram Web Resource.
http://mathworld.wolfram.com/CubicSpline.html

A Code

A.1 Product distribution: Threshold values

```python
from math import gamma
from mpmath import meijerg
from time import time


def search_threshold(L_1, L_2, nu_1, nu_2, CFAR, delta, initial):
    # Function that searches for threshold value u for given CFAR with accuracy delta
    # Parameters in the K^2-distribution are
    # L_j = ENL for channel j
    # nu_j = Order parameter for channel j, j = 1,2
    step = initial
    u = initial
    cdf = 0.0 # cumulative distribution function
    while step >= delta: # binary search
        # calculate cumulative distribution function
        cdf=meijerg([[1],[]],[[L_1,L_2,nu_1,nu_2],[0]],u)/(gamma(L_1)*gamma(L_2)*gamma(nu_1)*gamma(nu_2))
        # print ("%8.20f" %(cdf))
        # print u
        step = step / 2.0
        a = 1.0
        b = 1.0
        if 1-cdf > CFAR:
            u += step
        else:
            u -= step
        # print step

    return [u, cdf]


def multi_threshold(ENL, nu, CFAR, delta, initial):
    # Function that searches for for multiple threshold values u for given CFAR with accuracy delta
    # Parameters in the K^2-distribution are
    # ENL = vector with possible ENL values
    # nu = vector with possible order parameter values
    # initial = initial guess for binary search
    # Remark: The threshold values are bounded by 2*initial.
    # Initial should be chosen sufficiently large such that non-normalized threshold is within interval [0, 2*initial]

    # n_j = length of vector j, j = ENL, nu
    n_nu = len(nu)
    n_ENL = len(ENL)
    u = [0] * (n_ENL * n_ENL * n_nu * n_nu)  # threshold values
    s = 0  # iteration parameter to keep track of process
    tmp_u = 0.0

```

```
    for i in xrange(0, n_ENL):  # iteration over smallest ENL
        for j in xrange(i, n_ENL):  # iteration over largest ENL
            for k in xrange(0, n_nu):  # iteration over smallest order parameter
                for l in xrange(k, n_nu):  # iteration over largest order parameter
                    index=(i*n_ENL*n_nu*n_nu)+(j*n_nu*n_nu)+(k*n_nu)+l
                    if u[index] > 0:  # Check
                        print 'Index error'
                    else:
                        [tmp_u, _]=search_threshold(ENL[i],ENL[j],nu[k],nu[l],CFAR,
                                          ENL[i]*ENL[j]*nu[k]*nu[l]*delta, initial)
                        tmp_u /=(ENL[i]*ENL[j]*nu[k]*nu[l])  # Normalize
                        # threshold value for symmetric parameters
                        u[i*n_ENL*n_nu*n_nu + j*n_nu*n_nu + k*n_nu + l] = tmp_u
                        u[i*n_ENL*n_nu*n_nu + j*n_nu*n_nu + l*n_nu + k] = tmp_u
                        u[j*n_ENL*n_nu*n_nu + i*n_nu*n_nu + k*n_nu + l] = tmp_u
                        u[j*n_ENL*n_nu*n_nu + i*n_nu*n_nu + l*n_nu + k] = tmp_u
                    s += 1
                    print('Threshold value no. %d calculated. %d iterations remain'
                        %(s, int(len(ENL)*(len(ENL)+1)*len(nu)*(len(nu)+1)/4)-s))
    return u


def save_threshold(ENL, nu, CFAR, u):
    # Function that saves threshold values u for given CFAR with parameters ENL, nu in text-file
    # ENL = vector with possible ENL values
    # nu = vector with possible order parameter values

    n_nu = len(nu)
    n_ENL = len(ENL)

    file = open('threshold_CFAR_' + str(1 + CFAR)[2:] + '.txt', 'w')
    for i in xrange(0, (n_ENL * n_ENL * n_nu * n_nu) - 1):
        file.write(str(u[i]) + '\n')
    file.write(str(u[-1]))
    file.close()

    file = open('ENL.txt','w')
    for j in xrange(0,n_ENL-1):
        file.write(str(ENL[j]) + '\n')
    file.write(str(ENL[-1]))
    file.close()

    file = open('nu.txt', 'w')
    for k in xrange(0, n_nu-1):
        file.write(str(nu[k]) + '\n')
    file.write(str(nu[-1]))
    file.close()
    return 0


def check(ENL, nu, CFAR, u):
```

```
    # Function that serves as a quality check for the calculated threshold values u
    # It checks that the values for given CFAR are indexed correctly according to their corresponding parameters ENL, nu

    n_ENL = len(ENL)
    n_nu = len(nu)

    for i in xrange(0, n_ENL):  # iteration over ENL channel 1
        for j in xrange(0, n_ENL):  # iteration over ENL channel 2
            for k in xrange(0, n_nu):  # iteration over order parameter channel 1
                for l in xrange(0, n_nu):  # iteration over order parameter channel 2
                    index=(i *n_ENL*n_nu*n_nu)+(j*n_nu*n_nu)+(k*n_nu)+l
                    cdf = meijerg([[1],[]],[[ENL[i],ENL[j],nu[k],nu[l]],[0]],
                                 u[index]*ENL[i]*ENL[j]*nu[k]*nu[l])
                    cdf /= (gamma(ENL[i])*gamma(ENL[j])*gamma(nu[k])*gamma(nu[l])) # Normalize
                    if 1.0 - cdf > 1.01 * CFAR or 1.0 - cdf < 0.99*CFAR:
                        print index
                        return -1
    return 0

t0 = time()  # Register start time

ENL = [1.0, 2.0, 3.0, 4.0]  # ENL
nu = [5.0, 10.0, 15.0, 20.0, 40.0, 90.0]  # Order parameter
CFAR = 0.0000001  # Constant false alarm rate
delta = 0.00000001  # Accuracy
initial = 4000000.0  # Initial guess

u = multi_threshold(ENL, nu, CFAR, delta, initial)

print save_threshold(ENL, nu, CFAR, u)

print check(ENL, nu, CFAR, u)

t1 = time()  # Register stop time

print ('script takes %f seconds' %(t1-t0))  # print time
```

A.2 K-distribution: Threshold values

```
from math import gamma
from decimal import *
from mpmath import meijerg
from time import time
getcontext().prec = 100

def search_threshold(L, nu, CFAR, delta, initial):
    # Function that searches for threshold value u for given CFAR with accuracy
        delta
    # Parameters in the K-distribution are
    # L = ENL
    # nu = Order parameter for channel
    step = initial
    u = initial
    cdf = 0.0 # cumulative distribution function
    while step >= delta: # binary search
        # calculate cumulative distribution function
        cdf = meijerg([[1],[]],[[L, nu],[0]], u)/(gamma(L)*gamma(nu))
        # print ("%8.20f" %(cdf))
        # print u
        step = step / 2.0
        a = 1.0
        b = 1.0
        if 1-cdf > CFAR:
            u += step
        else:
            u -= step
        # print step

    return [u, cdf]


def multi_threshold(ENL, nu, CFAR, delta, initial):
    # Function that searches for for multiple threshold values u for given CFAR
        with accuracy delta
    # Parameters in the K-distribution are
    # ENL = vector with possible ENL values
    # nu = vector with possible order parameter values
    # n_j = length of vector j, j = ENL, nu
    # initial = initial guess for binary search
    # Remark: The threshold values are bounded by 2*initial.
    # Initial should be chosen sufficiently large such that non-normalized
       threshold is within interval [0, 2*initial]
    # Do some numerical "experiments" in advance

    n_ENL = len(ENL)
    n_nu = len(nu)
    u = [0] * (n_ENL * n_nu)  # threshold values
    s = 0  # iteration parameter to keep track of process
    tmp_u = 0.0
    tmp_cdf = 0.0
    for i in xrange(0, n_ENL):  # iteration over ENL
        for k in xrange(0, n_nu):  # iteration over order parameter
            index = (i * n_nu) + k
```

```
            if u[index] > 0:  # Check
                print 'Index error'
            else:
                [tmp_u, tmp_cdf] = search_threshold(ENL[i], nu[k], CFAR,
                                        ENL[i] * nu[k] * delta, initial)
                tmp_u = tmp_u/(ENL[i] * nu[k])  # Normalize
                u[i * n_nu + k] = tmp_u
                # print u
                s += 1
                print ('Threshold value no. %d calculated. %d iterations remain
                    '
                       %(s, int(n_ENL*n_nu)-s))
    return u


def save_threshold(ENL, nu, CFAR, u):
    # Function that saves threshold values u for given CFAR with parameters ENL
        , nu in text-file
    # ENL = vector with possible ENL values
    # nu = vector with possible order parameter values
    # n_j = length of vector j, j = ENL, nu
    n_ENL = len(ENL)
    n_nu = len(nu)

    file = open('threshold_CFAR_' + str(1 + CFAR)[2:] + '.txt', 'w')
    for i in xrange(0, (n_ENL* n_nu) - 1):
        file.write(str(u[i]) + '\n')
    file.write(str(u[-1]))
    file.close()

    file = open('ENL.txt','w')
    for j in xrange(0,n_ENL-1):
        file.write(str(ENL[j]) + '\n')
    file.write(str(ENL[-1]))
    file.close()

    file = open('nu.txt', 'w')
    for k in xrange(0, n_nu-1):
        file.write(str(nu[k]) + '\n')
    file.write(str(nu[-1]))
    file.close()
    return 0


def check_index(ENL, nu, CFAR, u):
    # Function that serves as a quality check for the calculated threshold
        values u.
    # It checks that the values for given CFAR are indexed correctly according
        to their corresponding parameters ENL, nu
    n_ENL = len(ENL)
    n_nu = len(nu)
    for i in xrange(0, n_ENL):  # iteration over ENL
        for k in xrange(0, n_nu):  # iteration over order parameter
            index = (i * n_nu) + k
            cdf = meijerg([[1],[]],[[ENL[i],nu[k]],[0]], u[index]*ENL[i]*nu[k])
            cdf /= (gamma(ENL[i]) * gamma(nu[k]))  # Normalize
            if 1.0 - cdf > 1.01 * CFAR or 1.0 - cdf < 0.99*CFAR:
```

```
                print index
                return -1
    return 0

t0 = time()  # Register start time

ENL = [1.0,2.0,3.0,4.0]  # ENL
nu = [5.0, 10.0, 15.0, 20.0, 40.0, 90.0]  # Order parameter
CFAR = 0.00000001
delta = 0.00000001
initial = 4000.0

u = multi_threshold(ENL, nu, CFAR, delta, initial)

print save_threshold(ENL, nu, CFAR, u)

print check_index(ENL, nu, CFAR, u)

t1 = time()  # Register stop time

print ('script takes %f seconds' %(t1-t0))  # print time
```

B Look-up tables

B.1 Product distribution, CFAR: 0.0000001

ENL_1	ENL_2	ν_1	ν_2	t
1.0	1.0	5.0	5.0	188.15227365
			10.0	159.05825320
			15.0	148.64651498
			20.0	143.25519888
			40.0	134.88170346
			90.0	130.05162249
		10.0	10.0	133.49937582
			15.0	124.33138640
			20.0	119.57610234
			40.0	112.17483841
			90.0	107.89387475
		15.0	15.0	115.59841859
			20.0	111.06462417
			40.0	103.99992501
			90.0	99.90745104
		20.0	20.0	106.64322469
			40.0	99.74853126
			90.0	95.75062921
		40.0	40.0	93.10932653
			90.0	89.25218007
		90.0	90.0	85.47142905

Product distribution, CFAR: 0.0000001

ENL_1	ENL_2	v_1	v_2	t
1.0	2.0	5.0	5.0	129.01381057
			10.0	108.10120831
			15.0	100.60521456
			20.0	96.71847642
			40.0	90.67101218
			90.0	87.17428865
		10.0	10.0	89.88773960
			15.0	83.34415536
			20.0	79.94537857
			40.0	74.64533502
			90.0	71.57169335
		15.0	15.0	77.13476711
			20.0	73.90648301
			40.0	68.86604028
			90.0	65.93807118
		20.0	20.0	70.76488707
			40.0	65.85581706
			90.0	63.00104337
		40.0	40.0	61.14448040
			90.0	58.39871690
		90.0	90.0	55.71204243

Product distribution, CFAR: 0.0000001

ENL_1	ENL_2	ν_1	ν_2	t
1.0	3.0	5.0	5.0	107.33152875
			10.0	89.44589900
			15.0	83.02770250
			20.0	79.69675624
			40.0	74.50772255
			90.0	71.50239395
		10.0	10.0	73.94772823
			15.0	68.37344829
			20.0	65.47536332
			40.0	60.95015896
			90.0	58.32108015
		15.0	15.0	63.09584890
			20.0	60.34928988
			40.0	56.05506750
			90.0	53.55573095
		20.0	20.0	57.67985930
			40.0	53.50263666
			90.0	51.06852752
		40.0	40.0	49.50162390
			90.0	47.16464136
		90.0	90.0	44.88020517

Product distribution, CFAR: 0.0000001

ENL_1	ENL_2	ν_1	ν_2	t
1.0	4.0	5.0	5.0	95.85092007
			10.0	79.57040682
			15.0	73.72311140
			20.0	70.68631773
			40.0	65.95107486
			90.0	63.20506196
		10.0	10.0	65.51245418
			15.0	60.45176209
			20.0	57.81875024
			40.0	53.70326432
			90.0	51.30882968
		15.0	15.0	55.66795749
			20.0	53.17647132
			40.0	49.27688264
			90.0	47.00378779
		20.0	20.0	50.75707460
			40.0	46.96690609
			90.0	44.75482700
		40.0	40.0	43.34157466
			90.0	41.22030807
		90.0	90.0	39.14806912

Product distribution, CFAR: 0.0000001

ENL_1	ENL_2	v_1	v_2	t
2.0	2.0	5.0	5.0	87.51779746
			10.0	72.67131073
			15.0	67.34197253
			20.0	64.57510262
			40.0	60.26243230
			90.0	57.76268727
		10.0	10.0	59.85680048
			15.0	55.24653077
			20.0	52.84878861
			40.0	49.10277169
			90.0	46.92458954
		15.0	15.0	50.88939971
			20.0	48.62107309
			40.0	45.07256790
			90.0	43.00548579
		20.0	20.0	46.41869292
			40.0	42.97035585
			90.0	40.95920388
		40.0	40.0	39.67309503
			90.0	37.74535285
		90.0	90.0	35.86280982

Product distribution, CFAR: 0.0000001

ENL_1	ENL_2	ν_1	ν_2	t
2.0	3.0	5.0	5.0	72.34499486
			10.0	59.73761128
			15.0	55.20739432
			20.0	52.85332275
			40.0	49.17958747
			90.0	47.04656689
		10.0	10.0	48.91382744
			15.0	45.01589413
			20.0	42.98678435
			40.0	39.81257349
			90.0	37.96344869
		15.0	15.0	41.34097577
			20.0	39.42602334
			40.0	36.42627044
			90.0	34.67540796
		20.0	20.0	37.56931490
			40.0	34.65814153
			90.0	32.95679020
		40.0	40.0	31.88069966
			90.0	30.25321039
		90.0	90.0	28.66569867

Product distribution, CFAR: 0.0000001

ENL_1	ENL_2	ν_1	ν_2	t
2.0	4.0	5.0	5.0	64.31893919
			10.0	52.89845139
			15.0	48.79142648
			20.0	46.65579910
			40.0	43.31981368
			90.0	41.38033414
		10.0	10.0	43.13006118
			15.0	39.60943739
			20.0	37.77544430
			40.0	34.90356653
			90.0	33.22813229
		15.0	15.0	36.29592272
			20.0	34.56803670
			40.0	31.85846156
			90.0	30.27452107
		20.0	20.0	32.89433297
			40.0	30.26722722
			90.0	28.72940930
		40.0	40.0	27.76469851
			90.0	26.29568400
		90.0	90.0	24.86385364

Product distribution, CFAR: 0.0000001

ENL_1	ENL_2	ν_1	ν_2	t
3.0	3.0	5.0	5.0	59.57167672
			10.0	48.91015360
			15.0	45.07544683
			20.0	43.08105213
			40.0	39.96480084
			90.0	38.15231360
		10.0	10.0	39.80756906
			15.0	36.52648183
			20.0	34.81696988
			40.0	32.13926399
			90.0	30.57643003
		15.0	15.0	33.44104804
			20.0	31.83182070
			40.0	29.30759927
			90.0	27.83134893
		20.0	20.0	30.27382894
			40.0	27.82764103
			90.0	26.39506575
		40.0	40.0	25.49941116
			90.0	24.13203652
		90.0	90.0	22.79992705

Product distribution, CFAR: 0.0000001

ENL_1	ENL_2	ν_1	ν_2	t
3.0	4.0	5.0	5.0	52.81841961
			10.0	43.18818481
			15.0	39.72176694
			20.0	37.91769738
			40.0	35.09614216
			90.0	33.45282770
		10.0	10.0	34.99789327
			15.0	32.04348652
			20.0	30.50312109
			40.0	28.08792670
			90.0	26.67621608
		15.0	15.0	29.27027906
			20.0	27.82287601
			40.0	25.55010516
			90.0	24.21882164
		20.0	20.0	26.42301185
			40.0	24.22271417
			90.0	22.93202667
		40.0	40.0	22.13205356
			90.0	20.90198408
		90.0	90.0	19.70466543

Product distribution, CFAR: 0.0000001

ENL_1	ENL_2	ν_1	ν_2	t
4.0	4.0	5.0	5.0	46.74011917
			10.0	38.05862571
			15.0	34.93141171
			20.0	33.30278657
			40.0	30.75317475
			90.0	29.26617923
		10.0	10.0	30.70467164
			15.0	28.05010827
			20.0	26.66513791
			40.0	24.49140023
			90.0	23.21890621
		15.0	15.0	25.56305255
			20.0	24.26410457
			40.0	22.22230498
			90.0	21.02440481
		20.0	20.0	23.00919064
			40.0	21.03459375
			90.0	19.87437495
		40.0	40.0	19.16175513
			90.0	18.05782747
		90.0	90.0	16.98428935

B.2 Product distribution, CFAR: 0.00000001

ENL_1	ENL_2	ν_1	ν_2	t
1.0	1.0	5.0	5.0	267.16917503
			10.0	222.10669927
			15.0	205.96711459
			20.0	197.59907308
			40.0	184.57623387
			90.0	177.04228614
		10.0	10.0	183.26928551
			15.0	169.33002748
			20.0	162.09116036
			40.0	150.80181130
			90.0	144.25214667
		15.0	15.0	156.16493750
			20.0	149.32195402
			40.0	138.63734720
			90.0	132.42854296
		20.0	20.0	142.68084858
			40.0	132.30340245
			90.0	126.26669102
		40.0	40.0	122.39061961
			90.0	116.61192825
		90.0	90.0	110.97447901

Product distribution, CFAR: 0.00000001

ENL_1	ENL_2	v_1	v_2	t
1.0	2.0	5.0	5.0	179.61633031
			10.0	148.04928606
			15.0	136.72866675
			20.0	130.85225616
			40.0	121.69201461
			90.0	116.38047634
		10.0	10.0	121.07265494
			15.0	111.37845130
			20.0	106.33808371
			40.0	98.46381467
			90.0	93.88410303
		15.0	15.0	102.25800480
			20.0	97.51156201
			40.0	90.08718054
			90.0	85.76158692
		20.0	20.0	92.91527699
			40.0	85.71980686
			90.0	81.52270703
		40.0	40.0	78.87122173
			90.0	74.86698430
		90.0	90.0	70.96829555

Product distribution, CFAR: 0.00000001

ENL_1	ENL_2	ν_1	ν_2	t
1.0	3.0	5.0	5.0	147.72227076
			10.0	121.11341161
			15.0	111.56213893
			20.0	106.60013609
			40.0	98.85647702
			90.0	94.35913893
		10.0	10.0	98.48963259
			15.0	90.35240604
			20.0	86.11803346
			40.0	79.49494160
			90.0	75.63618725
		15.0	15.0	82.71481929
			20.0	78.73667832
			40.0	72.50624477
			90.0	68.86955268
		20.0	20.0	74.88957857
			40.0	68.85913430
			90.0	65.33481003
		40.0	40.0	63.13201340
			90.0	59.77639983
		90.0	90.0	56.51303236

Product distribution, CFAR: 0.00000001

ENL_1	ENL_2	v_1	v_2	t
1.0	4.0	5.0	5.0	130.88264742
			10.0	106.89699223
			15.0	98.28108440
			20.0	93.80218634
			40.0	86.80627784
			90.0	82.73813347
		10.0	10.0	86.57625022
			15.0	79.26218484
			20.0	75.45368112
			40.0	69.49109201
			90.0	66.01239415
		15.0	15.0	72.40863747
			20.0	68.83648151
			40.0	63.23637736
			90.0	59.96284381
		20.0	20.0	65.38528664
			40.0	59.96992363
			90.0	56.80024976
		40.0	40.0	54.83490287
			90.0	51.82116566
		90.0	90.0	48.89259159

Product distribution, CFAR: 0.00000001

ENL_1	ENL_2	ν_1	ν_2	t
2.0	2.0	5.0	5.0	119.59431990
			10.0	97.72499981
			15.0	89.87330647
			20.0	85.79303399
			40.0	79.42236779
			90.0	75.71977051
		10.0	10.0	79.19845016
			15.0	72.53384624
			20.0	69.06481441
			40.0	63.63625690
			90.0	60.47111767
		15.0	15.0	66.28910515
			20.0	63.03555966
			40.0	57.93754532
			90.0	54.95957155
		20.0	20.0	59.89231245
			40.0	54.96281266
			90.0	52.07964936
		40.0	40.0	50.28920278
			90.0	47.54857268
		90.0	90.0	44.88598986

Product distribution, CFAR: 0.00000001

ENL_1	ENL_2	ν_1	ν_2	t
2.0	3.0	5.0	5.0	97.78373943
			10.0	79.46915416
			15.0	72.88829039
			20.0	69.46575940
			40.0	64.11601717
			90.0	61.00169482
		10.0	10.0	64.03611805
			15.0	58.48004042
			20.0	55.58573974
			40.0	51.05121091
			90.0	48.40267472
		15.0	15.0	53.28699984
			20.0	50.57923225
			40.0	46.33121148
			90.0	43.84514309
		20.0	20.0	47.96705346
			40.0	43.86526195
			90.0	41.46157148
		40.0	40.0	39.98577229
			90.0	37.70600895
		90.0	90.0	35.49408147

Product distribution, CFAR: 0.00000001

ENL$_1$	ENL$_2$	ν_1	ν_2	t
2.0	4.0	5.0	5.0	86.27926617
			10.0	69.84431416
			15.0	63.93496098
			20.0	60.85979308
			40.0	56.04877998
			90.0	53.24448739
		10.0	10.0	56.04692499
			15.0	51.07658360
			20.0	48.48580825
			40.0	44.42305589
			90.0	42.04677456
		15.0	15.0	46.43927507
			20.0	44.01975494
			40.0	40.22028993
			90.0	37.99345942
		20.0	20.0	41.68806571
			40.0	38.02308121
			90.0	35.87207295
		40.0	40.0	34.56268748
			90.0	32.52577906
		90.0	90.0	30.55128492

Product distribution, CFAR: 0.00000001

ENL_1	ENL_2	ν_1	ν_2	t
3.0	3.0	5.0	5.0	79.66321110
			10.0	64.38506293
			15.0	58.89105563
			20.0	56.03159709
			40.0	51.55695707
			90.0	48.94773995
		10.0	10.0	51.58088080
			15.0	46.96797313
			20.0	44.56317433
			40.0	40.79115406
			90.0	38.58405865
		15.0	15.0	42.66770161
			20.0	40.42372880
			40.0	36.89908126
			90.0	34.83249023
		20.0	20.0	38.26227194
			40.0	34.86403402
			90.0	32.86876304
		40.0	40.0	31.65822086
			90.0	29.77034275
		90.0	90.0	27.94125356

Product distribution, CFAR: 0.00000001

ENL_1	ENL_2	ν_1	ν_2	t
3.0	4.0	5.0	5.0	70.11032523
			10.0	56.43729160
			15.0	51.51744632
			20.0	48.95529463
			40.0	44.94236119
			90.0	42.59931061
		10.0	10.0	45.02258442
			15.0	40.90788951
			20.0	38.76153740
			40.0	35.39180640
			90.0	33.41731982
		15.0	15.0	37.07921332
			20.0	35.08011954
			40.0	31.93712164
			90.0	30.09156424
		20.0	20.0	33.15664585
			40.0	30.12960838
			90.0	28.34954175
		40.0	40.0	27.27924451
			90.0	25.59784370
		90.0	90.0	23.97043555

Product distribution, CFAR: 0.00000001

ENL_1	ENL_2	ν_1	ν_2	t
4.0	4.0	5.0	5.0	61.58933963
			10.0	49.37605318
			15.0	44.97886852
			20.0	42.68756115
			40.0	39.09565738
			90.0	36.99564230
		10.0	10.0	39.22042914
			15.0	35.55762902
			20.0	33.64588491
			40.0	30.64173529
			90.0	28.87894743
		15.0	15.0	32.15601463
			20.0	30.37886182
			40.0	27.58216502
			90.0	25.93749108
		20.0	20.0	28.67088712
			40.0	25.98038626
			90.0	24.39575767
		40.0	40.0	23.45184271
			90.0	21.95773713
		90.0	5.0	36.99564230
			10.0	28.87894743
			15.0	25.93749108
			20.0	24.39575767
			40.0	21.95773713
		90.0	90.0	20.51316385

B.3 K-distribution, CFAR: 0.0000001

ENL	ν	t
1.0	5.0	32.33718280
	10.0	25.07230963
	15.0	22.39907320
	20.0	20.98216068
	40.0	18.70552302
	90.0	17.32270455
2.0	5.0	20.26196923
	10.0	15.48611416
	15.0	13.72773347
	20.0	12.79431262
	40.0	11.28962713
	90.0	10.36957284
3.0	5.0	15.91029020
	10.0	12.04443735
	15.0	10.62039115
	20.0	9.86361959
	40.0	8.64081759
	90.0	7.88943467
4.0	5.0	13.61673051
	10.0	10.23368141
	15.0	8.98693594
	20.0	8.32380369
	40.0	7.25027873
	90.0	6.58796957

B.4 K-distribution, CFAR: 0.00000001

ENL	ν	t
1.0	5.0	39.60740649
	10.0	30.12173544
	15.0	26.63936878
	20.0	24.79370922
	40.0	21.82379504
	90.0	20.01185789
2.0	5.0	24.41911334
	10.0	18.32492583
	15.0	16.08750363
	20.0	14.90040391
	40.0	12.98519745
	90.0	11.80994710
3.0	5.0	18.98241587
	10.0	14.11755826
	15.0	12.33121919
	20.0	11.38269880
	40.0	9.84947302
	90.0	8.90456299
4.0	5.0	16.12805424
	10.0	11.91261168
	15.0	10.36448206
	20.0	9.54191774
	40.0	8.21023452
	90.0	7.38663564

About FFI

The Norwegian Defence Research Establishment (FFI) was founded 11th of April 1946. It is organised as an administrative agency subordinate to the Ministry of Defence.

FFI's MISSION

FFI is the prime institution responsible for defence related research in Norway. Its principal mission is to carry out research and development to meet the requirements of the Armed Forces. FFI has the role of chief adviser to the political and military leadership. In particular, the institute shall focus on aspects of the development in science and technology that can influence our security policy or defence planning.

FFI's VISION

FFI turns knowledge and ideas into an efficient defence.

FFI's CHARACTERISTICS

Creative, daring, broad-minded and responsible.

Om FFI

Forsvarets forskningsinstitutt ble etablert 11. april 1946. Instituttet er organisert som et forvaltningsorgan med særskilte fullmakter underlagt Forsvarsdepartementet.

FFIs FORMÅL

Forsvarets forskningsinstitutt er Forsvarets sentrale forskningsinstitusjon og har som formål å drive forskning og utvikling for Forsvarets behov. Videre er FFI rådgiver overfor Forsvarets strategiske ledelse. Spesielt skal instituttet følge opp trekk ved vitenskapelig og militærteknisk utvikling som kan påvirke forutsetningene for sikkerhetspolitikken eller forsvarsplanleggingen.

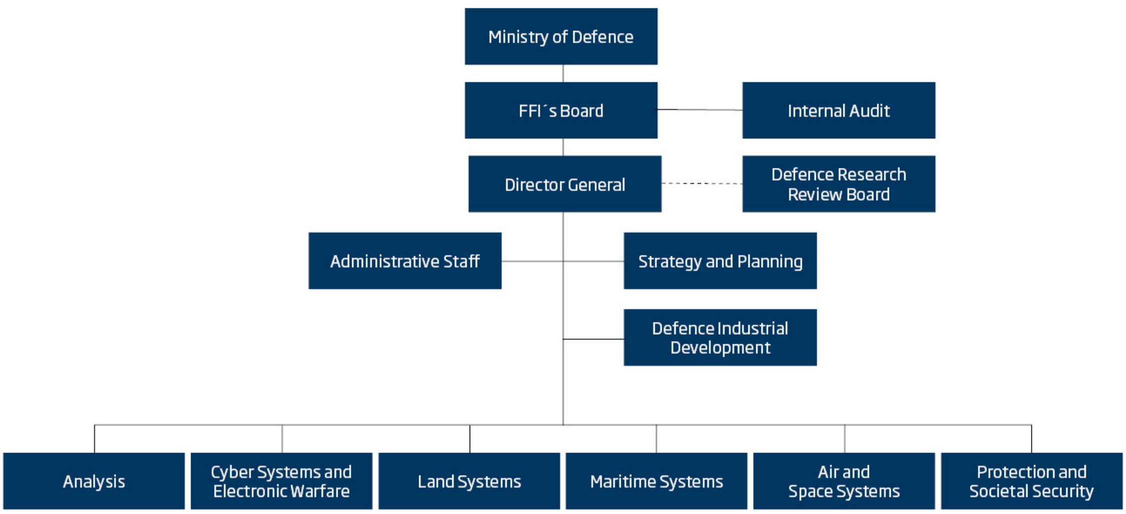
FFIs VISJON

FFI gjør kunnskap og ideer til et effektivt forsvar.

FFIs VERDIER

Skapende, drivende, vidsynt og ansvarlig.

FFI's organisation



Forsvarets forskningsinstitutt
Postboks 25
2027 Kjeller

Besøksadresse:
Instituttveien 20
2007 Kjeller

Telefon: 63 80 70 00
Telefaks: 63 80 71 15
Epost: ffi@ffi.no

Norwegian Defence Research Establishment (FFI)
P.O. Box 25
NO-2027 Kjeller

Office address:
Instituttveien 20
N-2007 Kjeller

Telephone: +47 63 80 70 00
Telefax: +47 63 80 71 15
Email: ffi@ffi.no

FFI Forsvarets forskningsinstitutt
Norwegian Defence Research Establishment