2021

Sensing Methods for Two-Target and Four-Target Detection in Time-Constrained Vector Poisson and Gaussian Channels

Muhammad Fahad
Michigan Technological University, mfahad@mtu.edu

SENSING METHODS FOR TWO-TARGET AND FOUR-TARGET DETECTION IN TIME-CONSTRAINED VECTOR POISSON AND GAUSSIAN CHANNELS

By

Muhammad Fahad

A DISSERTATION

Submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

In Electrical Engineering

MICHIGAN TECHNOLOGICAL UNIVERSITY

2021

This dissertation has been approved in partial fulfillment of the requirements for the Degree of DOCTOR OF PHILOSOPHY in Electrical Engineering.

Department of Electrical and Computer Engineering

Dissertation Advisor:	*Dr. Daniel R. Fuhrmann*
Committee Member:	*Dr. Timothy J. Schulz*
Committee Member:	*Dr. Zhaohui Wang*
Committee Member:	*Dr. Alexander E. Labovsky*
Department Chair:	*Dr. Glen E. Archer*

Dedication

To Manahil & Mustafa

Contents

List of Figures

List of Symbols

$I(X;Y)$	mutual information between random vectors X and Y, $0 \leq I(X;Y) \leq H(X)$
P_d	Bayesian probability of total correction detections, $0 \leq P_d \leq 1$
X	input random vector, $X \in \{\lambda_0, \lambda_1\}^N$
Y	output observation random vector, $Y \in \mathbb{Z}_+^M$ for Poisson channel and $Y \in \mathbb{R}^M$ for Gaussian channel
N	length of vector X
W	multivariate Gaussian noise vector
N_i	i^{th} component of noise vector W
λ_0	Poisson intensity parameter, $0 \leq \lambda_0 < \lambda_1$
λ_1	Poisson intensity parameter, $\lambda_1 > \lambda_0$
T	scalar, total available time resource, $T \geq 0$
p	prior probability of X_i, $0 \leq p \leq 1$
X_i	i^{th} component of $N-$dimensional input random vector X
Y_i	i^{th} component of $M-$dimensional output random vector Y
$Y\|X$	channel or conditional random vector Y given X
T_i	i^{th} component of counting-time proportion
$H(X)$	Shannon entropy of random vector X, $H(X) \geq 0$

$\mathcal{H}_i$	i^{th} hypothesis
$E[U]$	expected value of the random variable U
$\mathcal{Y}$	alphabet set of random vector Y
$\mathcal{X}$	alphabet set of random vector X
$\mathrm{Poiss}(U;\lambda)$	pmf of Poisson distributed random variable U with parameter λ
C_d	empirically computed probability of total correct detections, $0 \leq C_d \leq 1$
$\frac{d}{dT} I(X;Y)\Big\|_{T=0}$	derivative of mutual information I w.r.t T when evaluated at $T = 0$
$\mathcal{N}(w;\mu,\Sigma)$	multivariate Gaussian distribution with mean vector μ and covariance Σ, $= (2\pi)^{-\frac{k}{2}} \mathtt{det}(\Sigma)^{-\frac{1}{2}} e^{-\frac{1}{2}(w-\mu)^\intercal \Sigma^{-1}(w-\mu)}$, w might be omitted for brevity
$\mathcal{N}(y;\mu,\sigma^2)$	Gaussian distribution of scalar random variable Y at $Y = y$ with mean μ and variance σ^2
$P(Y)$	probability distribution of Y
$\mathrm{Log}_2[x]$	logarithm of x with base 2
r	Bayes risk
$P_{kl \mid ij}$	probability that $X = [\lambda_i, \lambda_j]^\intercal$ is true while decision $X = [\lambda_k, \lambda_l]^\intercal$ is made
γ	free parameter variable, $0 \leq \gamma$
α	scalar parameter, $0 \leq \alpha \leq T$
$\wedge$	logical AND operation

$=$	mathematical equality
$:=$	assignment operator from RHS to LHS
$\neq$	mathematical inequality
f_{ij}	conditional pmf of random vector Y when $X = [\lambda_i, \lambda_j]^\intercal$ occurs
:	such that
T_i^O	optimal value of time T_i
$\mathcal{T}$	dummy variable in the integration
I^O	optimal value of mutual information
D_i	decision i
$\{\mathcal{P}_i(t), t \geq 0\}$	i^{th} conditional point Poisson process
MC	Monte Carlo
$\sim$	is distributed as
Φ	scaling matrix, $\Phi \in \mathbb{R}_+^{M \times N}$
I	Identity matrix

Preface

Chapter 1 discusses sensor management for vector Poisson and Gaussian channels, and presents a review of the relevant literature.

Chapter 2 considers the two target detection problem in vector Poisson channel under a time constraint using metrics of mutual information I and Bayes probability of total correct detections P_d. For all computations, multivariate Poisson mixture pmfs are first truncated and then computations of I and P_d are performed for any input time argument.

Chapter 3 considers the two target detection problem in vector Gaussian channel under a time constraint using metrics of I and P_d. Monte Carlo method is used for all computations.

Chapter 4, investigates the various heuristic sensing schemes for the detection of four targets in vector Poisson and Gaussian channels under a time constraint. Since looking for an optimal solution requires a $15-$dimensional space to work with and that is not feasible for our approach, a few sub-optimal sensing methods are studied. Based on the observations, we have found analytical evidence that are mentioned in the form of theorems in this chapter that confirm that observations. For instance, concavity of I is observed in plots for both constrained and unconstrained objectives

under both channels and later we found in the literature that they have to be concave, thus confirming the computed results. Again Monte Carlo method is used for computing both I and P_d for any given input arguments.

Chapter 5 concludes the research findings and the future work.

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my advisor Dr. Daniel R. Fuhrmann for all his time and support during my PhD research.

Thank you to the committee members, Dr. Timothy J. Schulz, Dr. Zhaohui Wang, Dr. Alexander E. Labovsky for volunteering your time and being the part of my dissertation committee.

I feel gratitude towards Dr. Jeffrey B. Burl from whom I learnt *Detection and Estimation Theory* in EE 5521. This research was not possible without his insightful yet challenging lectures.

I am thankful to IPS staff, ECE staff, graduate school staff and registrar office staff.

A big thank you to the Fulbright fellowship program-2011 of the United States Department of State for awarding me a PhD scholarship at Michigan Tech.

I also want to thank my parents, siblings, family & friends for all their love, affection and support.

Last, but not the least I am heartily thankful to the people of the beautiful Upper Peninsula of Michigan.

Abstract

In this dissertation we consider a sensor scheduling or resource management problem for a vector Poisson and Gaussian channels. The input is a binary random vector and the output is a set of conditionally independent Poisson or Gaussian random variables. The objective is to design a scaling matrix, which is a linear transformation whose purpose is to entangle the different inputs, under a total given energy/time constraint. The two metrics are adopted to quantify the performance of the designed scaling matrix: *mutual information* and *Bayesian inference*. In other words, it is an experimental design problem where the objective is to glean the *information* about the binary inputs and perform the *classification* of the input random vector in a fixed time-resource, that is transmitted through a vector Poisson and Gaussian channel, based on the output observations.

No *optimal solution* is claimed in this dissertation for the above problem for either of the Poisson or Gaussian channels; from either of the two perspectives: mutual information or Bayes detection. However, time-symmetry does exist in the said problem. It is further noted that the problem is concave in its domain (i.e. sensing times) from mutual information criterion; and this is based on the observations in the computational results. If this concavity *does* exist in the problem then together with the *time-symmetry* result; it can be deduced that the *optimal solution* has a

symmetry too; and that would reduce the exponentially rising dimensionality of the search-space to the linear one (w.r.t dimension of the input random vector). However, concavity of the objective function in the Bayes framework does not exist.

Further, it is noted that the classification criterion in the above two channels; and mutual information criterion do not *generally* lead to the same solution when subjected to the same fixed time constraint and model parameters.

It is also noted that the combinatorial explosion is inevitable, that occurs while addressing the problem through computational means, even with exploiting the inherent *time-symmetry* and the *concavity* in the objective. This curse of the dimensionality is the main obstacle in exploring the problem for targets greater than four (i.e. for dimension of the input vector > 4).

Chapter 1

Introduction

The importance of sensor scheduling emerges when it is not possible to collect all the data all the time or when resources are limited [2]. This creates a need for an optimal or heuristic scheduling methodology to extract the data from different available data sources in an efficient manner. If the available resources are unlimited then it is always advantageous to collect all available data; however constraints imposed by available time, computational resources, memory requirements, communication bandwidth, available battery power, deployment pattern make this infeasible. One of the major hurdles in sensor scheduling arises from the combinatorial nature of the possible switching possibilities available to collect data. This problem becomes further complicated if the sensor scheduling is done in continuous-time settings [3].

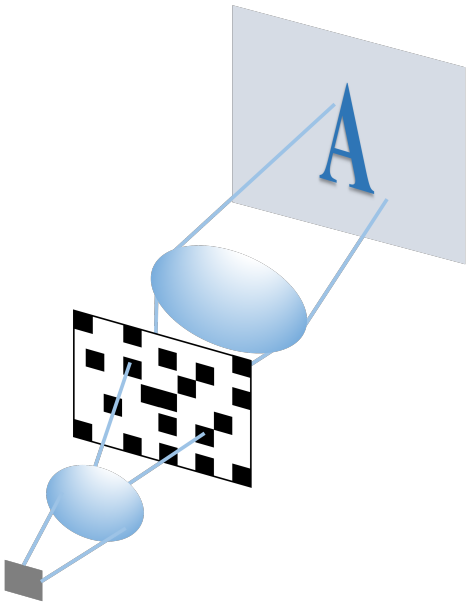

Figure 1.1: A single pixel camera. Grid of shutters control the spatial distribution of incoming light focusing on sensor.

Information-theoretic approaches have long been used in sensor scheduling problems, especially for their invariance to invertible transformations of the data. Many of the early approaches towards sensor management were information-theoretic, for instance [4], [5]. Maximizing the mutual information between the unknowns and the observations is a natural and intuitive approach to sensor resource management.

In this dissertation we address a problem of *scheduling* the available sensors for optimal or suboptimal time allocation to detect a binary 2-long and 4-long random vector in vector The Poisson channel and Gaussian channels with dark current noise λ_0. This might be of special interest in context of Poisson *compressive sensing*, where sparse signal conditions can be exploited for reduced computational cost

and efficient sensor scheduling. We address the problem of scheduling a single sensor from both *information theoretic* and *detection theoretic* perspectives. Poisson channel has remained traditionally difficult to work with due to the two basic inherent obstacles attached with it. First, added Poisson noise and scaling of input does not consolidate into a single parameter as SNR does in Gaussian channel. Second, scaling the support of Poisson random variable does not result in another Poisson random variable. This is in contrast to Gaussian channel which are relatively well-studied from both information-theoretic and estimation-theoretic aspects [6], [7], [8]. In an unconstrained sensing-time setting, vector Poisson channel has gained much research activity and interesting results are shown paralleling to that of vector Gaussian channel. One of the seminal works in linking the *information theory* and *estimation theory* in the context of scalar Poisson channel is done by Guo *et al.* in [9], where the derivative of mutual information with respect to signal intensity is equated with a function of conditional expectation; providing a ground for the possible Poisson counterpart of a Gaussian channel. This result for the scalar Poisson channel was further refined by Atar and Weissman in [10] where an exact relationship between derivative of mutual information and minimum mean loss error is given by providing the loss function $l(x, \hat{x}) = x \log(\frac{x}{\hat{x}}) - x + \hat{x}$ which shares a key property with squared error loss i.e., optimality of conditional expectation with respect to mean loss criterion. This result together with $\frac{d}{d\alpha} I(X; \mathcal{P}(\alpha X))\Big|_{\alpha=0} = E[X \cdot \operatorname{Log} X] - E[X] \cdot \operatorname{Log}(E[X])$ given in [9] provides us an exact answer to the

question: for a given finite sensing time T and prior p, which of the two sensing mechanisms, individual or joint sensing, is better than the other? (however, hybrid sensing still remains elusive and this is we have investigated in this chapter). Wang *et al.* [11] unifies the vector Poisson and Gaussian channels by constructing a generalization of the classical Bregman divergence and extended the scalar result to the vector case (unconstrained). They provide the gradient of mutual information with respect to their input scaling matrices for both Poisson and Gaussian channel. But, for a general vector Poisson channel existence of gradient of mutual information w.r.t scaling matrix is defined in terms of expected value of the Bregman divergence matrix with a strictly K-convex loss function and which requires the partial ordering interpretation [12] [11] and which does not unify our problem. We are therefore also interested in exploring the general concave nature of our problem w.r.t scaling matrix (if it exists), which is discussed in coming chapters.

For a Two-Target detection in a vector Poisson channel as illustrated in fig. (1.2), let $\{\mathcal{P}_1(t), t \geq 0\}$ and $\{\mathcal{P}_2(t), t \geq 0\}$be two *conditional point Poisson processes* [13, pp. 88-89] such that their rate parameters are defined by X_1 and X_2, respectively. We count the number of arrivals from the two *conditional Poisson arrival processes* in three possible configurations: two by individually counting the arrivals from the two processes $\mathcal{P}_1$, $\mathcal{P}_2$ and one by counting the arrivals from the sum of two processes $\{\mathcal{P}_1(t) + \mathcal{P}_2(t), t \geq 0\}$ with given rate parameter $X_1 + X_2$. The counting is performed in a manner that at any given time, only one of the three possible configurations

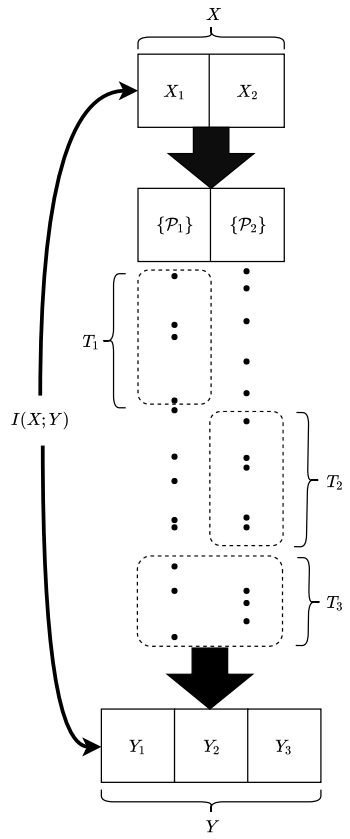

Figure 1.2: Illustration of sensing paradigm for detection of $2-$long hidden random vector X from $3-$long observable random vector Y in vector Poisson channel when scaling matrix is $\begin{bmatrix} T_1 & 0 \\ 0 & T_2 \\ T_3 & T_3 \end{bmatrix}$ and time constraint is $T = T_1+T_2+T_3$.

is active. Additionally, counting is performed in given finite time constraint $T = T_1 + T_2 + T_3$ where T_1, T_2 and T_3 be the unknown time proportions in counting arrivals from processes $\{\mathcal{P}_1(t), t \geq 0\}$, $\{\mathcal{P}_2(t), t \geq 0\}$ and $\{\mathcal{P}_1(t) + \mathcal{P}_2(t), t \geq 0\}$, respectively and T is given total time resource. After utilizing available time T, the above counting paradigm leads us to a multivariate Poisson mixture model for Y.

For a $4-$Target detection in a vector Poisson channel. The problem is set up such that there is a $4-$long binary input vector $X = [X_1, X_2, X_3, X_4]$ such that each X_i is a

discrete random variable that assumes either of the two known values: λ_0 or λ_1 with probability $(1-p)$ and p, respectively. All X_i are mutually independent and identically distributed. Conditioned on X_i, a Poisson process $\mathcal{P}_i(t)$ is initiated in continuous time t. It is known that If we count the arrivals for time T_i from the conditional Poisson process we have another conditional counting Poisson process whose rate parameter at instant T_i is $(T_i \cdot x_i)$. Hence we have, initially, four conditional Poisson processes: $\mathcal{P}_1(T_1 \cdot x_1)$; $\mathcal{P}_2(T_2 \cdot x_2)$; $\mathcal{P}_3(T_3 \cdot x_3)$ and $\mathcal{P}_4(T_4 \cdot x_4)$ depending on the realization that input vector X assumes. Let $\binom{4}{2}$ be the set containing all possible pairs constituted from four elements of X. Summing elements of each of the $6-$pairs we then have another six conditional Poisson processes: $\mathcal{P}_5(T_5 \cdot (x_1 + x_2))$; $\mathcal{P}_6(T_6 \cdot (x_1 + x_3))$; $\mathcal{P}_7(T_7 \cdot (x_1 + x_4))$; $\mathcal{P}_8(T_8 \cdot (x_2 + x_3))$; $\mathcal{P}_9(T_9 \cdot (x_2 + x_4))$ and $\mathcal{P}_{10}(T_{10} \cdot (x_3 + x_4))$. Considering $\binom{4}{3}$, we have four processes: $\mathcal{P}_{11}(T_{11} \cdot (x_1 + x_2 + x_3))$; $\mathcal{P}_{12}(T_{12} \cdot (x_1 + x_2 + x_4))$; $\mathcal{P}_{13}(T_{13} \cdot (x_1 + x_3 + x_4))$ and $\mathcal{P}_{14}(T_{14} \cdot (x_2 + x_3 + x_4))$. Summing all the four components of X, we have $\mathcal{P}_{15}(T_{15} \cdot (x_1 + x_2 + x_3 + x_4))$. Hence, there are $15-$ conditional point processes (in total) that we have to deal with to extract the maximum possible information or perform the best input signal detection by setting the counting times from T_1 to T_{15} in a fixed given time $\sum_{i=1}^{15} = T$ as illustrated in fig. (1.3).

The ideal way to address the problem would be to search for a solution in a $15-$ dimensional search-space by allowing $(T_1, T_2, \cdots T_{15})$ to have fifteen degrees-of-freedom. However, due to the computational complexity involved in exploring all fifteen dimensions we restrict ourselves to a reduced dimensional search-space.

Therefore, we have considered only some special cases of time-configurations. Four different types of time configurations are studied for each of the channels. We call: *individual* sensing when total given time T is equally divided into $T_1 = T_2 = T_3 = T_4 = \frac{T}{4}$; *pair-wise* sensing when $T_5 = T_6 = T_7 = T_8 = T_9 = T_{10} = \frac{T}{6}$; *triplets* sensing when $T_{11} = T_{12} = T_{13} = T_{14} = \frac{T}{4}$ and *joint* sensing when $T_{15} = T$.

For a four target detection, we consider the vector Poisson channel model [11]:

$$\begin{aligned} \mathrm{Pois}(Y; \Phi X) = P_{Y|X}(Y|X) &= \prod_{i=1}^{15} P_{Y_i|X}(Y_i|X) \\ &= \prod_{i=1}^{15} \mathrm{Pois}(Y_i; (\Phi X)_i) \end{aligned} \tag{1.1}$$

where $\mathrm{Pois}(U; z)$ denotes the standard Poisson distribution of random variable U with parameter z.

We consider the vector Gaussian channel model as defined in [11] i.e., $Y|X \sim \mathcal{N}(\Phi X, \mathrm{I})$, where $\mathcal{N}(\Phi X, \mathrm{I})$ is a multivariate Gaussian distribution with mean vector ΦX and unit covariance matrix I. The sensing mechanism is such that every combination of input random vector X is passed first through an integrate-and-dump receiver and then independent of the input, a Gaussian noise of unit variance is added into the signal. We assume this noise as some measurement noise, which is constant and independent of the signal magnitude, or the number of signals which are combined. A sensing mechanism is illustrated in fig.(1.4) where for each of the

possible 15 combinations of 4 components of X, there is a separate integrate and dump receiver and then a fixed Gaussian noise term is added.

The problem is to design Φ, for both channels, satisfying the time constraint $\sum_{i=1}^{15} T_i = T$. Note that the Φ matrices are different for the two channels as given in chapter 4. The mutual information I and Bayes probability of total correct detections P_d is then computed for any given sensing scheme and respective time-proportion constraint.

We start from detection problem of the two targets in chapter 2 and chapter 3 for Poisson and Gaussian channels respectively, and then move to the problem of four targets detection in chapter 4.

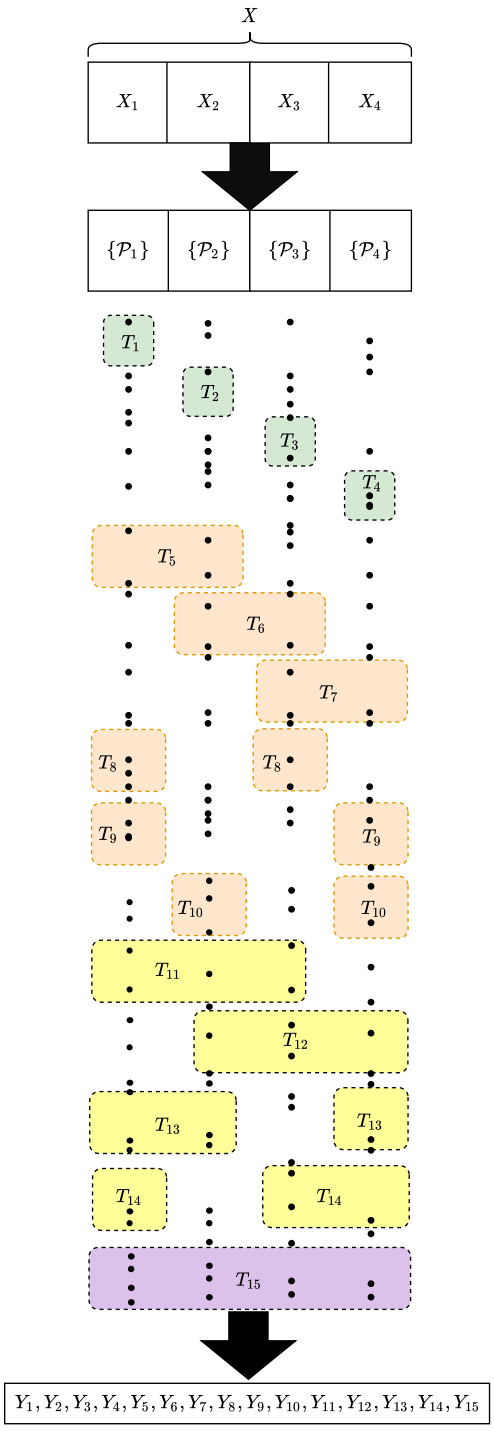

Figure 1.3: Illustration of sensing paradigm for detection of $4-$long hidden random vector X from $15-$long observable random vector Y through a vector Poisson channel under a total time constraint of $T = \sum_{i=1}^{15} T_i$. Each X_i is i.i.d and $P(X_i = \lambda_0) = 1 - p$ and $P(X_i = \lambda_1) = p$.

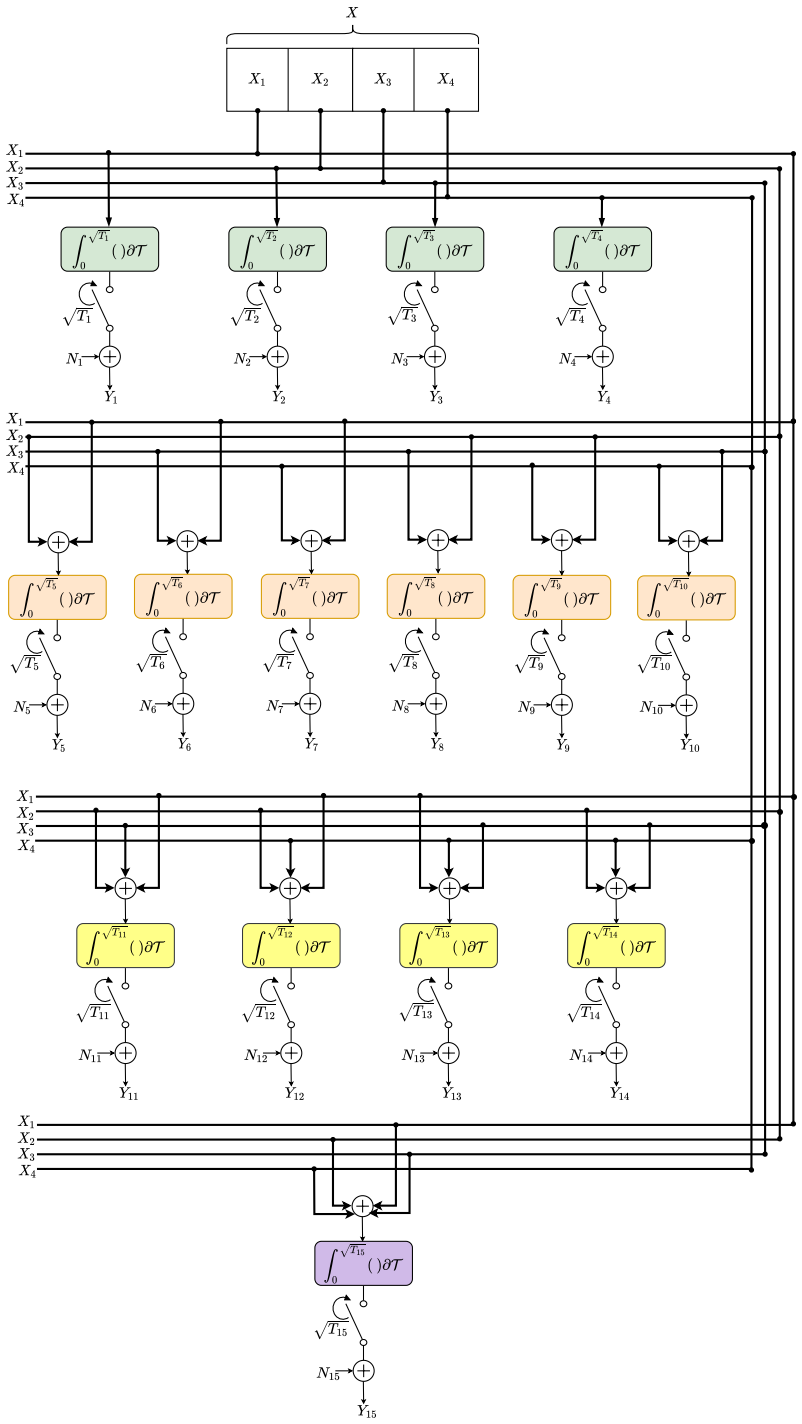

Figure 1.4: Sensing paradigm for detection of $4-$long random vector X from $15-$long random vector Y through a vector Gaussian channel under a time constraint of $T = \sum_{i=1}^{15} T_i$. Each Y_i is the sum of: output of integrate and dump receiver; and $N_i \sim \mathcal{N}(\mu = 0, \sigma^2 = 1)$. Each N_i is i.i.d Gaussian random variable. Each X_i is i.i.d and $P(X_i = \lambda_0) = 1 - p$ and $P(X_i = \lambda_1) = p$.

Chapter 2

Sensing Method for Two-Target Detection in Time-Constrained Vector Poisson Channel

2.1 Problem Description

Let X_1 and X_2 be two independent and identical distributed (i.i.d) Bernoulli random variables with p being the probability of occurrence of 1. We may define probability

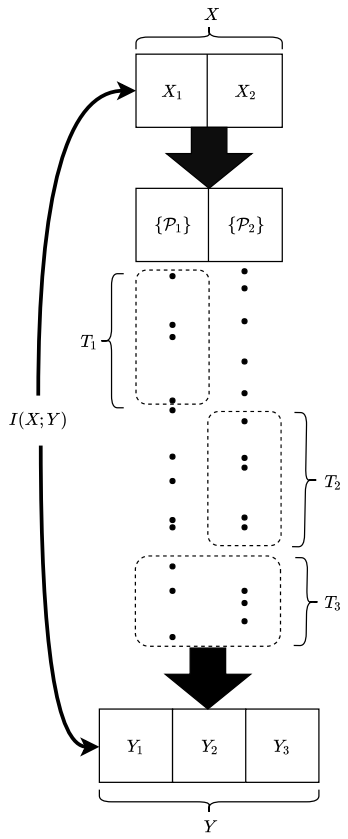

Figure 2.1: Illustration of sensing paradigm for detection of $2-$long hidden random vector X from $3-$long observable random vector Y in vector Poisson channel when scaling matrix is $\begin{bmatrix} T_1 & 0 \\ 0 & T_2 \\ T_3 & T_3 \end{bmatrix}$ and total time constraint is $T = T_1 + T_2 + T_3$.

mass function f of discrete random vector $X \equiv [X_1, X_2]^\intercal$ as

$$f_X(x) = \begin{cases} p^2 & x = [1 \quad 1]^\intercal \\ (1-p)^2 & x = [0 \quad 0]^\intercal \\ p(1-p) & x = [0 \quad 1]^\intercal \quad \text{or} \quad x = [1 \quad 0]^\intercal \end{cases} \tag{2.1}$$

Let $\{\mathcal{P}_1(t), t \geq 0\}$ and $\{\mathcal{P}_2(t), t \geq 0\}$be two *conditional point Poisson processes* [13, pp. 88-89] such that their rate parameters are defined by X_1 and X_2, respectively. We

may count the number of arrivals from the two *conditional Poisson arrival processes* in three possible configurations: two by individually counting the arrivals from the two processes $\mathcal{P}_1$, $\mathcal{P}_2$ and one by counting the arrivals from the sum of two processes $\{\mathcal{P}_1(t) + \mathcal{P}_2(t), t \geq 0\}$ with given rate parameter $X_1 + X_2$ as illustrated in Fig.(2.1). The counting is performed such that at any given time, only one of the three possible configurations is active. Furthermore, because of the *independent increments* property of *conditional Poisson processes*, it is not necessary to switch back and forth among possible configurations; it is sufficient to be in configuration 1 for time T_1, followed by configuration 2 for time T_2 then configuration 3 for time T_3. Additionally, counting is performed in given finite time constraint $T = T_1 + T_2 + T_3$ where T_1, T_2 and T_3 be the unknown time proportions in counting arrivals from processes $\{\mathcal{P}_1(t), t \geq 0\}$, $\{\mathcal{P}_2(t), t \geq 0\}$ and $\{\mathcal{P}_1(t) + \mathcal{P}_2(t), t \geq 0\}$, respectively and T is given total time resource. After utilizing available time T, the above counting paradigm leads us to a multivariate Poisson mixture model with four component in three dimensions. We write random vector $Y^\intercal \equiv [Y_1 \ Y_2 \ Y_3]$, so that $Y \in \{0, 1, 2, 3 \cdots\}^3$. Each Y_i is a Poisson random variable given X such that their conditional law is,

$$\begin{aligned} Y_1|X_1 &\sim \text{Poiss}\Big(y_1; \big(\lambda_0 \cdot (1 - X_1) + \lambda_1 \cdot X_1\big) \cdot T_1\Big) \\ Y_2|X_2 &\sim \text{Poiss}\Big(y_2; \big(\lambda_0 \cdot (1 - X_2) + \lambda_1 \cdot X_2\big) \cdot T_2\Big) \\ Y_3|(X_1 + X_2) &\sim \text{Poiss}\Big(y_3; \big(2 - (X_1 + X_2) \cdot \lambda_0 + (X_1 + X_2) \cdot \lambda_1\big) \cdot T_3\Big) \end{aligned} \quad (2.2)$$

where $\mathrm{Poiss}(y_i;\lambda_i) \equiv \frac{e^{-\lambda_i}(\lambda_i)^{y_i}}{y_i}$. The observable random vector Y carries *information* about hidden random vector $X^\intercal \equiv [X_1\ X_2]$. We are interested in finding if there exist any circumstance, in terms of available finite time T and known prior p, in which any of the three possible sensing methods: joint sensing, individual sensing or hybrid sensing is advantageous over the other.

In matrix form we may relate the rates of conditional Poisson distributed Y_i's as,

$$\begin{aligned}\overbrace{\begin{bmatrix} T_1 \cdot X_1 \\ T_2 \cdot X_2 \\ T_3 \cdot (X_1 + X_2) \end{bmatrix}}^{\Lambda:\,3\times1} &= \overbrace{\begin{bmatrix} T_1 & 0 & 0 \\ 0 & T_2 & 0 \\ 0 & 0 & T_3 \end{bmatrix}}^{\mathcal{D}:\,3\times3} \overbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}}^{\mathcal{B}:\,3\times2} \overbrace{\begin{bmatrix} X_1 \\ X_2 \end{bmatrix}}^{X:\,2\times1} \\ &= \overbrace{\begin{bmatrix} T_1 & 0 \\ 0 & T_2 \\ T_3 & T_3 \end{bmatrix}}^{\Phi:\,3\times2} \overbrace{\begin{bmatrix} X_1 \\ X_2 \end{bmatrix}}^{X:\,2\times1}.\end{aligned} \tag{2.3}$$

From the optimization point of view and in terms of *sensor scheduling*, we are interested in finding the optimal time-allocation, (T_1, T_2, T_3), of total available time resource, T, that would maximize the *reward* i.e. either the mutual information or probability of total correct detections. We may say that we are interested in finding the diagonal matrix $\mathcal{D}$, such that the mutual information $I(X;Y)$ is maximized under

time constraint $T = T_1 + T_2 + T_3$. Configuration matrix $\mathcal{B}$ represents all possible sensing combinations and Λ is the Poisson rate parameter vector. Mathematically we may write

$$\max_{T_1,T_2,T_3} I(X_1, X_2; Y_1, Y_2, Y_3) \text{ s.t. } T_1 + T_2 + T_3 = T. \tag{2.4}$$

We may rewrite the objective function

$$\max_{\alpha_1,\alpha_2,\alpha_3} I(X_1, X_2; Y_1, Y_2, Y_3) \text{ s.t. } \alpha_1 + \alpha_2 + \alpha_3 = 1. \tag{2.5}$$

Since parameters in the models λ_0,λ_1 and T can always be consolidated into $\lambda_0 \cdot T$ and $\lambda_1 \cdot T$, and by assuming $T = 1$ does preserve the problem statement equivalent to (2.4) with new parameters $\lambda_0 \cdot T$ and $\lambda_1 \cdot T$.

We extend our understanding by looking at the same problem from the *detection theoretic* aspect. For this, we maximize the Bayesian probability of total correct detection, P_d, of hidden random vector X from observable random vector Y, as

$$\max_{T_1,T_2,T_3} P_d \quad \text{s.t.} \quad T_1 + T_2 + T_3 = T \tag{2.6}$$

and compare the results to formerly computed *information theoretic* results. Simulations on empirical data are performed by varying different parameters involved in the model for further validation of computed results.

From now on, we use shorthand of $f(X)$ and $\mathrm{Poiss}(x;\theta)$ to represent $f_X(x)$ and $\frac{e^{\theta}\theta^{x}}{x!}$, respectively.

2.2 Information Theoretic Description

2.2.1 Scalar Poisson channel

Firstly the scalar version of the Poisson channel is presented and then it is extended to the vector version of our problem. We start with mutual information between a scalar random variable X_1 which is a transformed Bernoulli scalar random variable and Y_1 is a scalar Poisson mixture. The probability mass function of Y_1 is then given as

$$f(Y_1) = (1-p)\ \mathrm{Poiss}(y_1; T\lambda_0) + p\ \mathrm{Poiss}(y_1; T\lambda_1).$$

The mutual information I can be written as

$$I(X_1; Y_1) = H(Y_1) - H(Y_1|X_1)$$

where $H(\cdot)$ is the Shannon entropy, which is defined as a discrete functional $H : f \to -\sum_{Y\in\mathcal{Y}} f \cdot \mathrm{Log}_2(f)$ and f is the probability mass function of random variate Y

with $\mathcal{Y}$ as the corresponding support. We may write $H(Y_1)$ as

$$H(Y_1) = -\sum_{y_1=0}^{\infty} \Big([(1-p) \cdot \text{Poiss}(y_1; T\lambda_0) + p \cdot \text{Poiss}(y_1; T\lambda_1)] \cdot \text{Log}_2[(1-p) \cdot \text{Poiss}(y_1; T\lambda_0) + p \cdot \text{Poiss}(y_1; T\lambda_1)] \Big),$$

and

$$\begin{aligned} H(Y_1|X_1) &= \sum_{x_1 \in \mathcal{X}_1} f(x_1) \sum_{y_1=0}^{\infty} -f(y_1|x_1) \cdot \text{Log}_2[f(y_1|x_1)] \\ &= -\Big((1-p) \sum_{y_1=0}^{\infty} \text{Poiss}(y_1; T\lambda_0) \cdot \text{Log}_2[\text{Poiss}(y_1; T\lambda_0)] + p \sum_{y_1=0}^{\infty} \text{Poiss}(y_1; T\lambda_1) \cdot \text{Log}_2[\text{Poiss}(y_1; T\lambda_1)] \Big). \end{aligned}$$

Fig. (2.2) illustrates the concavity of mutual information with respect to time in the input intensity of a scalar Poisson channel when no time constraint is imposed. In the following section we formulate the mutual information expression for our vector Poisson channel shown in Fig. (2.1).

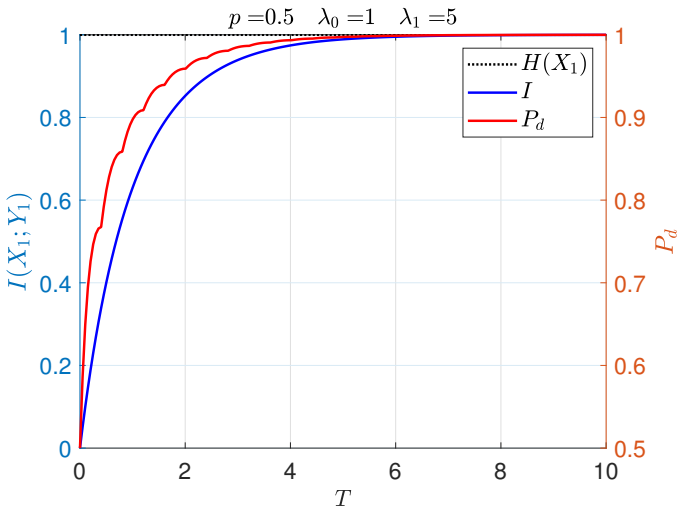

Figure 2.2: Mutual information $I(X_1;Y_1)$, entropy $H(X_1)$ and probability of total correct detections P_d vs. time T.
(MATLAB-CODE F.1)

2.2.2 Vector Poisson channel

Mutual information between two random vectors can be defined as the difference between the total entropy in one random vector and the conditional entropy in the second random vector given the first vector. We write

$$I(X;Y) = H(Y) - H(Y|X) \tag{2.7}$$

The conditional entropy $H(Y|X)$ is calculated from the conditional probability mass

functions $f(Y|X_i)$ defined as

$$
\begin{aligned}
& f(Y|X = [0 \quad 0]^\intercal) \\
&= \text{Poiss}(Y_1; \lambda_0 T_1) \cdot \text{Poiss}(Y_2; \lambda_0 T_2) \cdot \text{Poiss}(Y_3; 2\lambda_0 T_3), \\
& f(Y|X = [0 \quad 1]^\intercal) \\
&= \text{Poiss}(Y_1; \lambda_0 T_1) \cdot \text{Poiss}(Y_2; \lambda_1 T_2) \cdot \text{Poiss}(Y_3; (\lambda_0 + \lambda_1) T_3), \\
& f(Y|X = [1 \quad 0]^\intercal) \\
&= \text{Poiss}(Y_1; \lambda_1 T_1) \cdot \text{Poiss}(Y_2; \lambda_0 T_2) \cdot \text{Poiss}(Y_3; (\lambda_1 + \lambda_0) T_3), \\
& f(Y|X = [1 \quad 1]^\intercal) \\
&= \text{Poiss}(Y_1; \lambda_1 T_1) \cdot \text{Poiss}(Y_2; \lambda_1 T_2) \cdot \text{Poiss}(Y_3; 2\lambda_1 T_3).
\end{aligned}
$$

The marginal probability mass function of Y is then given as

$$
\begin{aligned}
& f(Y) \\
&= (1-p)^2 \cdot f(Y|X = [0 \quad 0]^\intercal) + p(1-p) \cdot \\
& f(Y|X = [0 \quad 1]^\intercal) + p(1-p) \cdot f(Y|X = [1 \quad 0]^\intercal) + \\
& p^2 \cdot f(Y|X = [1 \quad 1]^\intercal).
\end{aligned} \tag{2.8}
$$

Using the identity defined in (2.7) and the definition of entropy defined above, the

mutual information $I(X;Y)$ becomes

$$
\begin{aligned}
& I(X;Y) \\
&= \sum_{Y_1=0}^{\infty}\sum_{Y_2=0}^{\infty}\sum_{Y_3=0}^{\infty} -f(Y_1,Y_2,Y_3)\cdot \mathrm{Log}_2[f(Y_1,Y_2,Y_3)]) \\
&\quad - \sum_{X\in\mathcal{X}} f(X)\Big[\sum_{Y_1=0}^{\infty}\sum_{Y_2=0}^{\infty}\sum_{Y_3=0}^{\infty} -f(Y_1,Y_2,Y_3|X) \\
&\quad \cdot \mathrm{Log}_2[f(Y_1,Y_2,Y_3|X)]\Big]. \qquad (2.9)
\end{aligned}
$$

A complete expression for mutual information is given in Appendix A.

Theorem 2.1 *$I(X_1,X_2;Y_1,Y_2,Y_3)$ is concave function of T_1, T_2 and T_3 in $T_3=0$ plane.*

Proof:

$I(X_1,X_2;Y_1,Y_2,Y_3)\Big|_{T3=0} = I(X_1,X_2;Y_1,Y_2)$

By chain rule of mutual information:

$$
\begin{aligned}
I(X_1,X_2;Y_1,Y_2) &= I(X_1,X_2;Y_1) + I(X_1,X_2;Y_2|Y_1) \\
&= I(X_1;Y_1) + I(X_2;Y_2). \qquad (2.10)
\end{aligned}
$$

We note in (2.10) that each $I(X_i;Y_i)$ is solely a function of T_i and also concave in it [10, p. 1306]. We further know that sum of concave functions is a concave function. Therefore $I(X_1,X_2;Y_1,Y_2)$ is concave in T_1 and T_2 when $T_3=0$. This concludes the proof. ■

Theorem 2.2 *$I(X_1, X_2; Y_1, Y_2, Y_3)$ is symmetric in variables T_1 and T_2.*

Proof: Mutual information $I(X_1, X_2; Y_1, Y_2, Y_3)$ given in appendix A is invariant under any permutation of variables T_1 and T_2. That means interchanging the two variables leaves the expression unchanged. ■

If we further expand the expression for mutual information between X and Y using chain rule [14] [15] as,

$$\begin{aligned}
& I(X_1, X_2; Y_1, Y_2, Y_3) \\
&= I(X_1, X_2; Y_1) + I(X_1, X_2; Y_2|Y_1) + I(X_1, X_2; Y_3|Y_1, Y_2) \\
&= I(X_1; Y_1) + \overbrace{I(X_2; Y_1|X_1)}^{0} + \overbrace{I(X_1; Y_2|Y_1)}^{0} + \\
&\quad \overbrace{I(X_2; Y_2|Y_1, X_1)}^{I(X_2;Y_2)} + I(X_1, X_2; Y_3|Y_1, Y_2) \\
&= I(X_1; Y_1) + I(X_2; Y_2) + I(X_1, X_2; Y_3|Y_1, Y_2). \qquad (2.11)
\end{aligned}$$

Note that in (2.11) that the first and second terms are indeed concave in (T_1, T_2, T_3). The third term, when computed, exhibits non-concavity in the respective domain. Fig. (2.3) illustrates this when mutual information is plotted along the line $(T_1, T_2, T_3) := (\frac{T-T_3}{2}, \frac{T-T_3}{2}, T_3)$ parameterized by $0 \leq T_3 \leq T$. It is interesting to note that the sum of three terms exhibits concavity, irrespective of the fact that the third term is non-concave. However, analytical investigation of the functional properties of this third term, $I(X_1X_2; Y_3|Y_1Y_2)$, is not done in this work. Based on our observations,

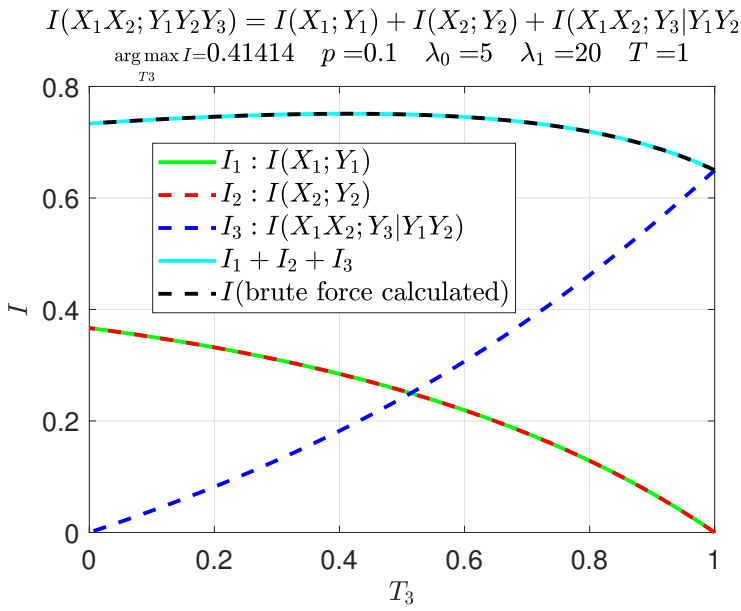

Figure 2.3: Concavity and convexity of three terms in $I(X_1X_2;Y_1Y_2Y_3) = I(X_1;Y_1) + I(X_2;Y_2) + I(X_1X_2;Y_3|Y_1Y_2)$ under symmetry and total time constraint $T_1 + T_2 + T_3 = T$.
(MATLAB-CODE F.2 & F.18.1)

with various values of model parameters, we would propose a conjecture.

Conjecture 2.1 *If $X \equiv [X_1\ X_2]^\intercal$ be a non-negative random vector where X_1 and X_2 are mutually independent and identical distributed. Let random vector $Y \equiv [Y_1\ Y_2\ Y_3]^\intercal$ be in non-negative integer space, jointly distributed with X such that conditional law is given as:*

$$
\begin{aligned}
Y_1|X &\sim \text{Poiss}(T_1 \cdot X_1) \\
Y_2|X &\sim \text{Poiss}(T_2 \cdot X_2) \\
Y_3|X &\sim \text{Poiss}\left(T_3 \cdot (X_1 + X_2)\right),
\end{aligned}
$$

where $T_1 \geq 0$*,* $T_2 \geq 0$ *and* $T_3 \geq 0$*, then* $I(X_1, X_2; Y_1, Y_2, Y_3)$ *is concave in* (T_1, T_2, T_3) *under time constraint* $T = T_1 + T_2 + T_3$*.*

Corollary 2.1 (Feasible region for optimal solution) *The two properties of mutual information* concavity *(if true) and* symmetry *(proved), guarantee that* maxima *must occur at the line of symmetry* $(T_1, T_2, T_3) := (\frac{T-\alpha}{2}, \frac{T-\alpha}{2}, \alpha)$ *where* $0 \leq \alpha \leq T$*.*

One of the immediate consequences of exploiting *symmetry* and *concavity* of the problem is that it would reduce the search space, from two dimensions to one dimension, of the original problem. *Claim* : Mutual information $I(X_1, X_2; Y_1, Y_2, Y_3)$ in general is not symmetric in p around $p = 0.5$ for fixed time proportions (T_1, T_2, T_3) for a conditionally multivariate Poisson (Y_1, Y_2, Y_3) given (X_1, X_2).

Example: For a scalar random variable X and conditional Poisson variable $\mathcal{P}(\alpha X)$ with scaling factor α, we have equation (115) on page (10) of [9], given below

$$\frac{d}{d\alpha} I(X; \mathcal{P}(\alpha X))\Big|_{\alpha=0} = E[X \cdot \operatorname{Log} X] - E[X] \cdot \operatorname{Log}(E[X]). \tag{2.12}$$

For our problem where discrete vector $[X_1, X_2]$ is distributed as:

$$f_X(x) = \begin{cases} p^2 & x \equiv [\lambda_1, \lambda_1]^\intercal \\ (1-p)^2 & x \equiv [\lambda_0, \lambda_0]^\intercal \\ p(1-p) & \text{otherwise} \end{cases} \tag{2.13}$$

with $T_1 = T_2 = \frac{T}{2}$ and $T_3 = 0$, we have

$$I(X_1, X_2; Y_1, Y_2, Y_3)\Big|_{\left(T_1=\frac{T}{2}, T_2=\frac{T}{2}, T_3=0\right)} = 2 \cdot I(X_1; Y_1)\Big|_{\left(T_1=\frac{T}{2}\right)} \tag{2.14}$$

Let T be a free variable in (2.14), we have

$$\frac{d}{dT}\left(I(X_1, X_2; Y_1, Y_2, Y_3)\Big|_{\left(T_1=\frac{T}{2}, T_2=\frac{T}{2}, T_3=0\right)}\right)\Bigg|_{T=0} = (1-p) \cdot \lambda_0 \cdot \log_2(\lambda_0) +$$
$$p \cdot \lambda_1 \cdot \log_2(\lambda_1) - \Big((1-p) \cdot \lambda_0 + p \cdot \lambda_1\Big) \cdot \log_2\Big((1-p) \cdot \lambda_0 + p \cdot \lambda_1\Big). \tag{2.15}$$

Equation (2.15) is concave in p

$$\frac{d^2}{dp^2}\left(\frac{d}{dT}\left(I(X_1, X_2; Y_1, Y_2, Y_3)\Big|_{\left(T_1=\frac{T}{2}, T_2=\frac{T}{2}, T_3=0\right)}\right)\Bigg|_{T=0}\right) =$$
$$-\frac{(\lambda_0 - \lambda_1)^2}{\mathrm{Ln}(2) \cdot \big((1-p)\lambda_0 + p \cdot \lambda_1\big)} < 0. \tag{2.16}$$

Maxima of equation (2.15) occurs at $p = \frac{4-e}{e}$ when $\lambda_0 = 2$ and $\lambda_1 = 4$. Since (2.15) is concave in p, then for being symmetric maxima should have occurred at $p = 0.5$.

■

2.3 Detection Theoretic Description

2.3.1 Maximization of probability of total correct detections in sensing time intervals

In previous section, we discussed and presented the metric of mutual information I between hidden random vector X, and observed vector Y. Here we approach the original sensing problem as a multi-hypothesis detection problem and find the optimal solution by minimizing the Bayesian risk [16, pp.220] in sensing times i.e., (T_1, T_2, T_3). We define Bayes risk r as

$$
\begin{aligned}
r = (1-p)^2 \Big[& P_{00\,|\,00}\, C_{00\,|\,00} + P_{01\,|\,00}\, C_{01\,|\,00} \\
& + P_{10\,|\,00}\, C_{10\,|\,00} + P_{11\,|\,00}\, C_{11\,|\,00} \Big] + p(1-p) \\
& \Big[P_{00\,|\,01}\, C_{00\,|\,01} + P_{01\,|\,01}\, C_{01\,|\,01} + P_{10\,|\,01}\, C_{10\,|\,01} \\
& + P_{11\,|\,01}\, C_{11\,|\,01} \Big] + p(1-p) \Big[P_{00\,|\,10}\, C_{00\,|\,10} \\
& + P_{01\,|\,10}\, C_{01\,|\,10} + P_{10\,|\,10}\, C_{10\,|\,10} + P_{11\,|\,10}\, C_{11\,|\,10} \Big] \\
& + p^2 \Big[P_{00\,|\,11}\, C_{00\,|\,11} + P_{01\,|\,11}\, C_{01\,|\,11} + P_{10\,|\,11}\, C_{10\,|\,11} \\
& + P_{11\,|\,11}\, C_{11\,|\,11} \Big],
\end{aligned}
$$

where $P_{kl\,|\,ij}$ is the probability that $X = [\lambda_i, \lambda_j]^\intercal$ is true while decision $X = [\lambda_k, \lambda_l]^\intercal$ is made; similarly for $C_{kl\,|\,ij}$. Setting all costs for which $[\lambda_i, \lambda_j]^\intercal \neq [\lambda_k, \lambda_l]^\intercal$ to one and $[\lambda_i, \lambda_j]^\intercal = [\lambda_k, \lambda_l]^\intercal$ to zero, we have

$$\begin{aligned} r = (1-p)^2 \Big[P_{01\,|\,00} \; + P_{10\,|\,00} \; + P_{11\,|\,00} \Big] + p(1-p) \\ \Big[P_{00\,|\,01} \; + P_{10\,|\,01} \; + P_{11\,|\,01} \Big] + p(1-p) \\ \Big[P_{00\,|\,10} \; + P_{01\,|\,10} \; + P_{11\,|\,10} \Big] + p^2 \\ \Big[P_{00\,|\,11} \; + P_{01\,|\,11} \; + P_{10\,|\,11} \Big]. \end{aligned} \tag{2.17}$$

We are interested in minimizing this Bayes risk r in (T_1, T_2, T_3) i-e

$$\min_{T_1, T_2, T_3} r \quad \text{s.t. } T_1 + T_2 + T_3 = T. \tag{2.18}$$

Note that while minimizing r in (T_1, T_2, T_3), the decisions boundaries would be changing accordingly and become function of (T_1, T_2, T_3). Equivalently, we may say that

$$\max_{T_1, T_2, T_3} P_d \quad \text{s.t. } T_1 + T_2 + T_3 = T \tag{2.19}$$

where P_d is probability of total correct detections, $P_d = 1 - r$. In the next section we presented the computed results of (2.19) and (2.9) and discuss how the two empirically relate to each other.

2.4 Computational and Simulation Results

The primary purpose of performing the computations and simulations is twofold: first is to see under what circumstances which of the sensing methods ,individual, joint or hybrid, is optimal; and second is to investigate the relationship that might exist between detection theory and information theory for a time constrained vector Poisson channel. A *possible* concave nature of the original problem, defined in (2.4) in the respective argument, (T_1, T_2, T_3), is noted when a pre-defined channel scaling matrix structure is imposed. Throughout our computations and simulations the input distribution of any i.i.d X_i is considered as Bernoulli random variable with probability of 1 be p, for decent comparison between different results and scenarios as channel parameters (T_1, T_2, T_3) are varied. For all computational purposes, we always approximate the Poisson mass function by truncating it; truncation is done by a rectangular window that extends from lower limit 0 to the upper limit where Poisson pmf drops in value below double precision machine epsilon ($\epsilon = 2^{-53}$) as given in Appendix C.

Since analytical solutions are impossible to achieve for our problem we may resort to numerical optimization techniques. We computed mutual information between hidden random vector X and observable random vector Y for diverse set of intensities λ_0 and λ_1 along with different priors and total available time, T, to observe

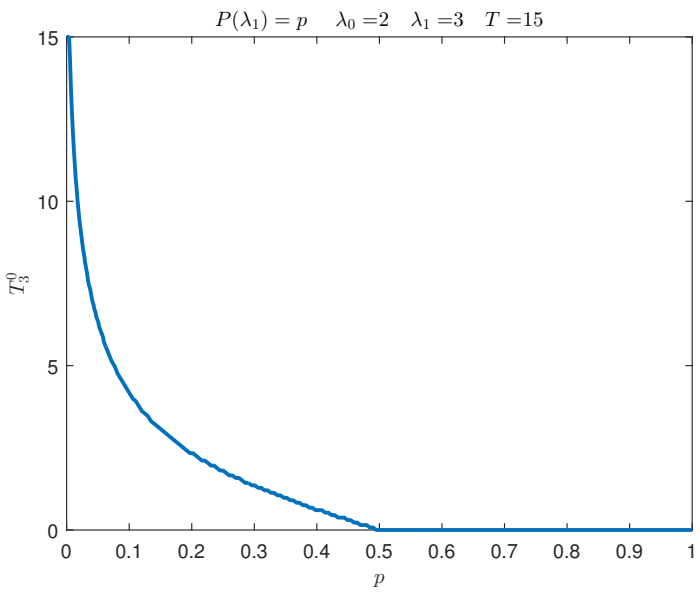

Figure 2.4: Optimal time T_3^0 vs. prior probability p with $T_1 = T_2$ and $T_1 + T_2 + T_3 = T$.
(MATLAB-CODE F.3 & F.18.2)

the concavity of $I(X_1, X_2; Y_1, Y_2, Y_3)$ in T_1, T_2 and T_3. We may employ convex optimization techniques by exploiting the concave nature of the objective function when $T_1 = T_2$. However, we don't have a proof that objective function is indeed concave in a linear time constraint. Concavity is based on our computational viewpoint as we did not succeed in catching a single case where non-concavity is observed. Based on our observation we consistently noted the concave property of mutual information in respective domain which led us to propose conjecture 2.1.

In Fig. (2.7) and Fig. (2.8), the input prior p is varied, the total available time T is held fixed while T_1, T_2 and T_3 are such that constraint $T_1 + T_2 + T_3 = T$ is always satisfied. It can be seen that the distribution of time resource changes as p is varied such that as p tends to be close to zero, the optimal value of T_3 tends to assume

all available time resource T leaving optimal values of T_1 and T_2 approaching to 0. Further, due to the inherent symmetry in the problem in arguments T_1 and T_2, plus considering the problem to be concave; the optimal point always varies along the line that bisects the constraint region (equilateral triangle) into two right angle triangles.

A relationship between information theory and detection theory for our problem is investigated in Fig. (2.9), in which Bayesian probability of total correct detections, P_d, given in Appendix B is plotted against respective mutual information I given in Appendix A. We further performed the simulation in Fig. (2.10) by generating conditional Poisson arrivals with given prior probability p and then count the arrivals. 10^4 random vector data samples Y from Poisson counting process are taken for every input of time proportions (T_1, T_2, T_3). A Bayesian detector is then employed to classify the incoming data/arrival-counts. The detector's decision are then finally compared with the actual inputs to determine the empirical total correct detections against each input time proportion. We found that the empirical total correct detections C_d is consistent with analytic P_d.

The computing method is described in the flowcharts in Fig.(D.1) and Fig.(D.2). The bounds on the entropy of Poisson variable are available in [1], we use them to validate that the computed Poisson entropies are within bounds.

The first derivative check at $T = 0$ given in [9] along with knowing the fact that

MI is concave in T [10] would be helpful in answering which of the two: joint or individual sensing is better than the other for a given finite time and Poisson intensities with given prior p.

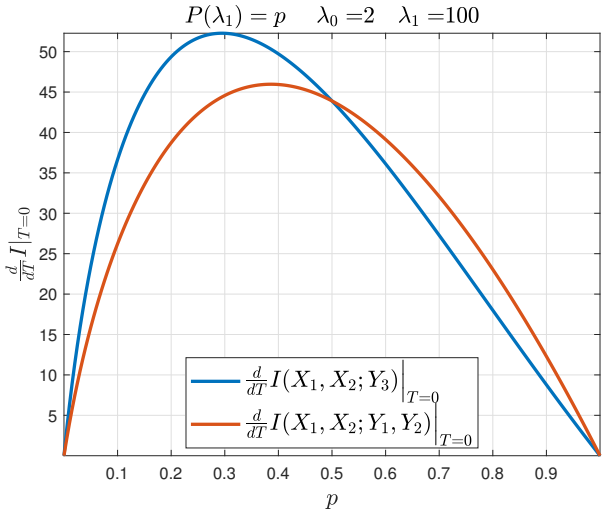

Figure 2.5: Individual 2-target detection problem: $\frac{d}{dT} I(X_1, X_2; Y_3)\Big|_{T=0}$ and $\frac{d}{dT} I(X_1, X_2; Y_1, Y_2)\Big|_{T=0}$ with $(T_1 = \frac{T}{2}, T_2 = \frac{T}{2})$ vs. p.
(MATLAB-CODE F.4)

For instance, in Fig.(2.5) we have $\frac{d}{dT} I(X_1, X_2; Y_3)\Big|_{T=0}$ and $\frac{d}{dT} I(X_1, X_2; Y_1, Y_2)\Big|_{T=0}$ vs. p for $\lambda_0 = 2$ and $\lambda_1 = 100$. We can see that at $p = 0.5$ the first derivatives of both the joint and individual sensing w.r.t time variable T are same; this means that no matter what the given time is, individual sensing is always better over the joint sensing. Whereas, for the two regions: $p > 0.5$ and $p < 0.5$ we need to further know what the given time is to decide which of the two methods is better over the other. This is because the two MI curves crosses each other as illustrated in Fig (2.6). Further note that in Fig. (2.6) the numerical derivative values of MI at

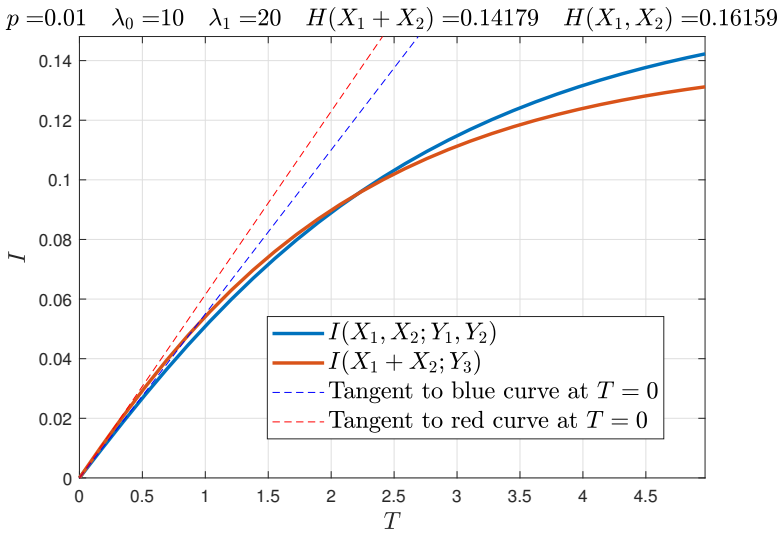

Figure 2.6: Joint sensing and individual sensing vs. T. Note that joint sensing is better sensing method when $T < 2.25$. For $T > 2.25$ individual sensing is better.
(MATLAB-CODE F.5)

$T = 0$ are 0.055011541474291 and 0.061490803759598 for individual and joint sensing, respectively when MI is computed according to the algorithm defined in flowcharts. Now compare these numerical derivative values with analytically calculated ones 0.055011540931595 and 0.061490807291534; the two are accurate to eight decimal places.

It is observed in Fig. (2.11) and Fig. (2.12) that the further the four conditional Poisson pmfs are from each other, the more the optimal solution relies on individual sensing provided the prior $p < 0.5$. In other words if the four pmfs (in the hypothesis testing) become closer, more the optimal sensing relies on joint sensing for the prior $p < 0.5$. It can be seen further that there exist an inverse relationship between the optimal mutual information $I^O(X;Y)$ value and its optimal argument α^O in the

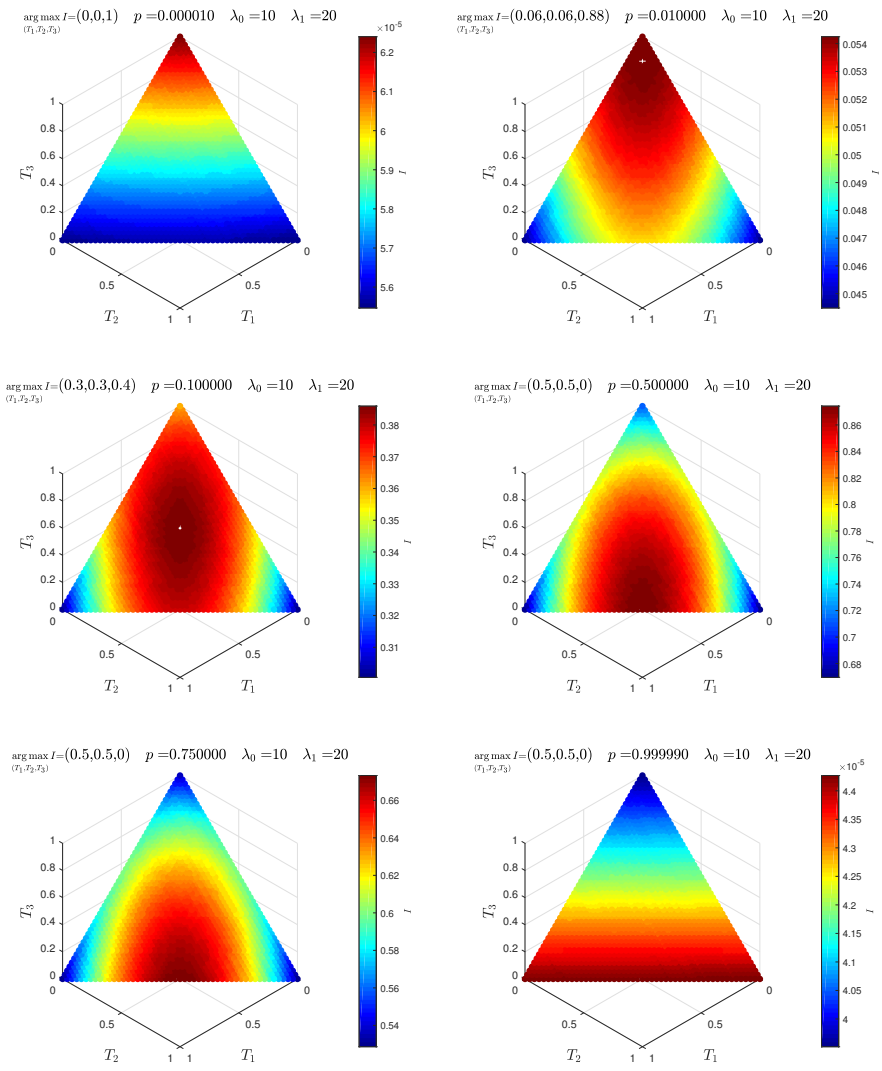

Figure 2.7: $I(X;Y)$ vs. (T_1, T_2, T_3) under time constraint $T_1 + T_2 + T_3 = 1$ for $\lambda_0 = 10$, $\lambda_1 = 20$, $T = 1$ and varying *prior* probability p. It can be seen from (a)-(f) that as prior p varies from 0.00001 to 0.5, the optimal solution drifts from $(0, 0, 1)$ to $(0.5, 0.5, 0)$ and stays there as p is varied further from 0.5 to 0.99999 along the line of symmetry $(T_1, T_2, T_3) := (\frac{1-\alpha}{2}, \frac{1-\alpha}{2}, \alpha)$ where $0 \leq \alpha \leq 1$.
(MATLAB-CODE F.6 & F.18.2)

region $\lambda_1 T > \lambda_0 T$. If the prior $p \geq 0.5$ then irrespective of the other parameter values in the model, individual sensing is the optimal strategy.

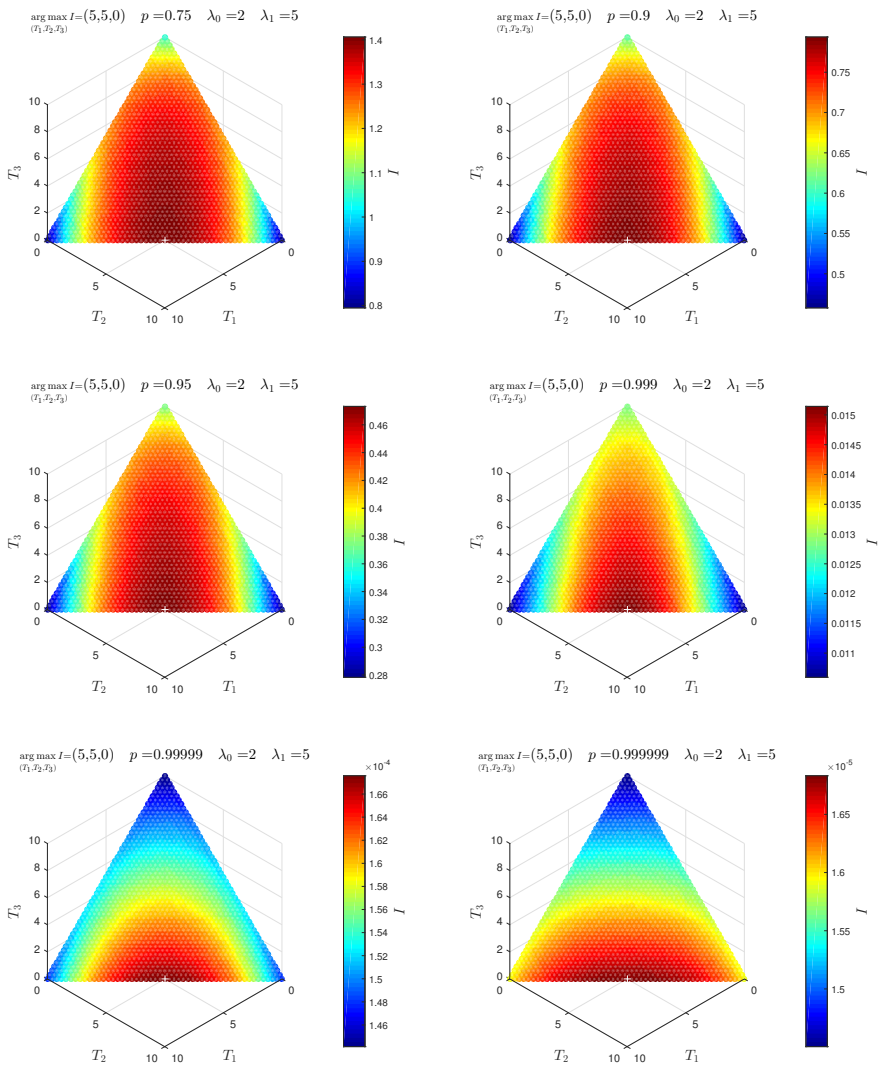

Figure 2.8: $I(X;Y)$ vs. (T_1, T_2, T_3) under time constraint $T_1 + T_2 + T_3 = T$ for $\lambda_1 = 5$, $\lambda_0 = 2$, $T = 10$ and varying *prior* probability p. It can be seen from (a)-(f) that as prior p moves from 0.5 to 0.999999, the optimal solution remains fixed at $(5, 5, 0)$.
(MATLAB-CODE F.6 & F.18.2)

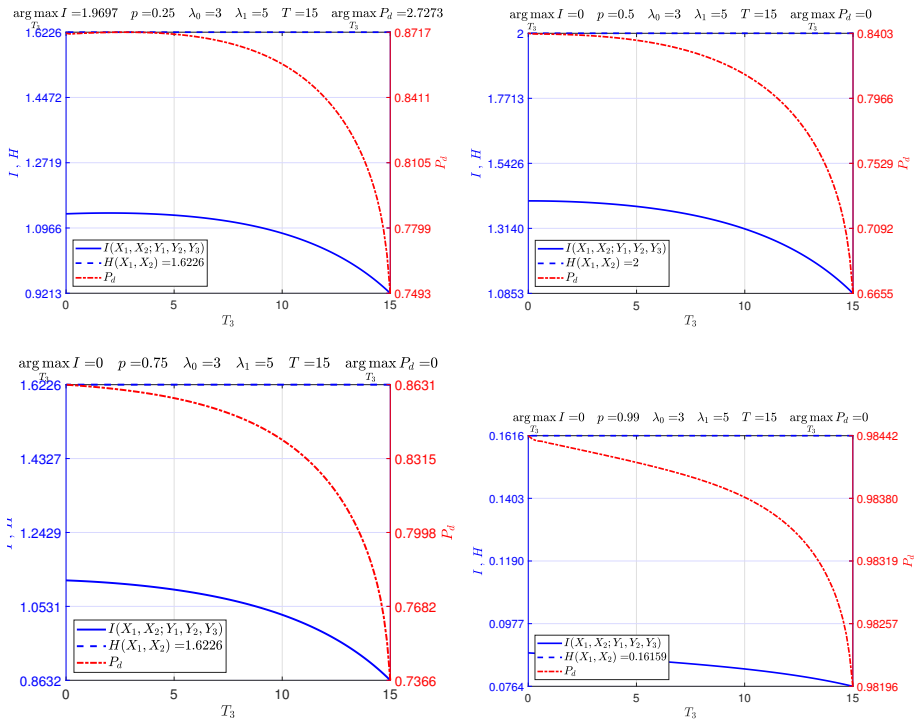

Figure 2.9: Mutual information $I(X;Y)$ and probability of total correct detections P_d vs. T_3 for *prior* probabilities of $0.25, 0.5, 0.75$ and 0.99. (MATLAB-CODE F.7 & F.18.2)

2.5 Conclusion

This work attempts to address the problem of sensor scheduling in a vector Poisson channel for a two target detection problem using criteria of mutual information and Bayesian risk with $0-1$ loss function. From non-decreasing and concave nature of mutual information w.r.t given finite time and knowing the first derivative of MI at $T = 0$, we can conclude that there definitely exist circumstances under which for a given time, joint sensing is advantageous over individual sensing. It is further observed that if the higher intensity of Poisson point process is more likely than

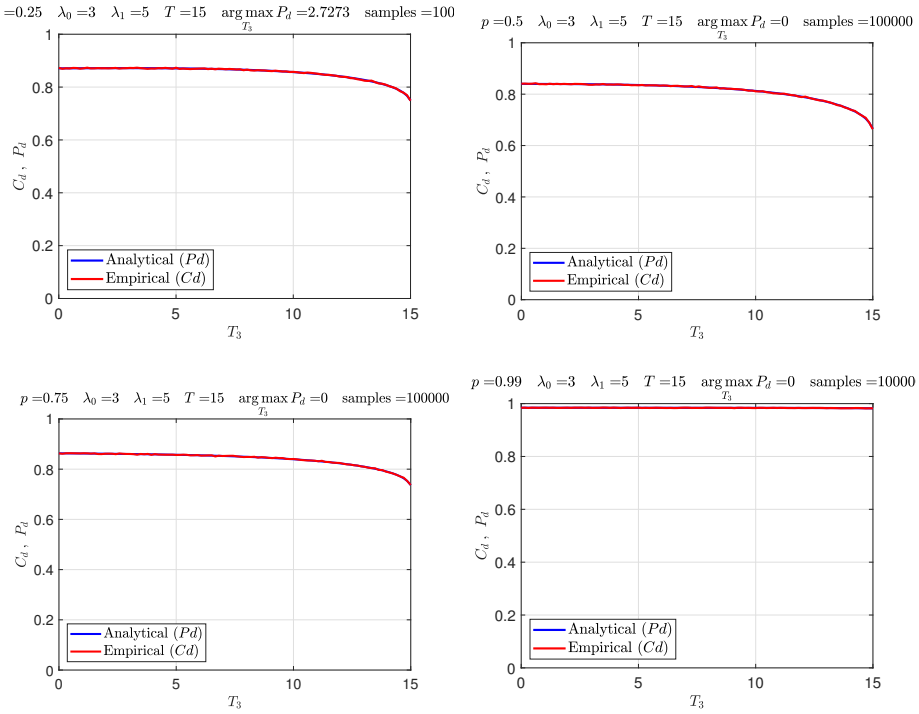

Figure 2.10: Empirical Correct-decision rate C_d and analytical probability of total correct detections P_d vs. T_3 for *prior* probabilities of $0.25, 0.5, 0.75$ and 0.99.
(MATLAB-CODE F.7 & F.18.2)

lower intensity $p \geq 0.5$ then optimal sensing method is to count the Poisson arrivals from the two targets individually; irrespective of the available time. Whereas, if the lower intensity is more likely $(1-p) \geq 0.5$ then it is the hybrid sensing that yields better results but it is computationally hefty to solve. The objective functions, I and P_d, are observed to be concave in sensing time under total time constraint, however no analytical evidence is presented. It would be interesting to know the optimal solution when more than two targets are present under similar conditions and does the concavity of mutual information and probability of total correct detections w.r.t sensing times still exist in higher dimensions.

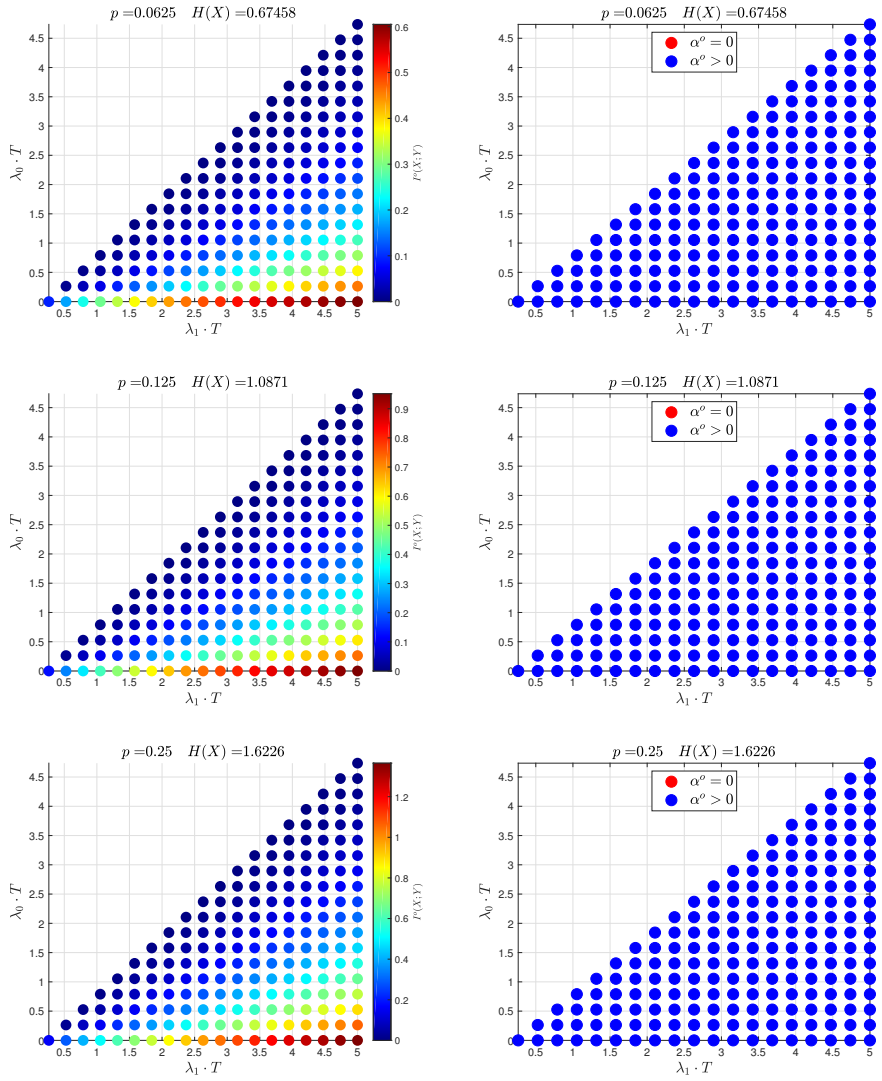

Figure 2.11: Left: $I^O(X;Y)$ vs. $(\lambda_0 T, \lambda_1 T)$ in the region $\lambda_1 T > \lambda_0 T$, right: corresponding optimal argument parameter α^O vs. $(\lambda_0 T, \lambda_1 T)$ for varying *prior* probabilities p. The search for each optimal argument α^O for any fixed: $(\lambda_0 T, \lambda_1 T)$ and p is performed over the line $(\alpha_1, \alpha_2, \alpha_3) := (\frac{1-\alpha}{2}, \frac{1-\alpha}{2}, \alpha)$ where $0 \leq \alpha \leq 1$ and $\alpha_1 + \alpha_2 + \alpha_3 = 1$.
(MATLAB-CODE F.8 & F.18.6)

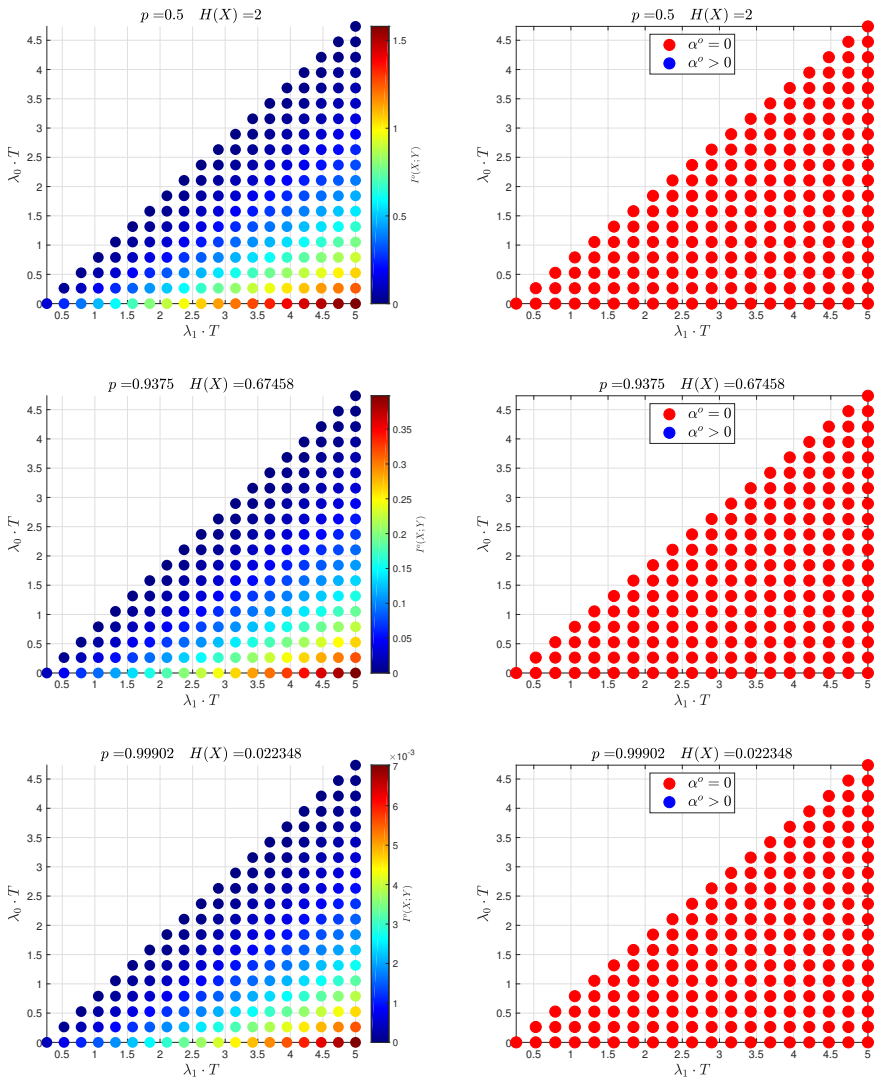

Figure 2.12: Left: $I^O(X;Y)$ vs. $(\lambda_0 T, \lambda_1 T)$ in the region $\lambda_1 T > \lambda_0 T$, right: corresponding optimal argument parameter α^O vs. $(\lambda_0 T, \lambda_1 T)$ for varying *prior* probabilities p. The search for each optimal argument α^O for any fixed: $(\lambda_0 T, \lambda_1 T)$ and p is performed over the line $(\alpha_1, \alpha_2, \alpha_3) := (\frac{1-\alpha}{2}, \frac{1-\alpha}{2}, \alpha)$ where $0 \leq \alpha \leq 1$ and $\alpha_1 + \alpha_2 + \alpha_3 = 1$.
(MATLAB-CODE F.8 & F.18.6)

Chapter 3

Sensing Method for Two-Target Detection in Time-Constrained Vector Gaussian Channel

This chapter deals with binary input signalling observed through a vector Gaussian channel. For almost two decades it is well-known that for an arbitrary distributed finite power input signal through either a scalar or vector Gaussian channel there exist a relationship that relates the mutual information (MI), between the input and output, and the minimum mean-square error (MMSE) achievable by optimal conditional estimation of input given the output and that is the derivative of MI w.r.t snr is equal to half the MMSE, irrespective of the input distribution. In this work, we

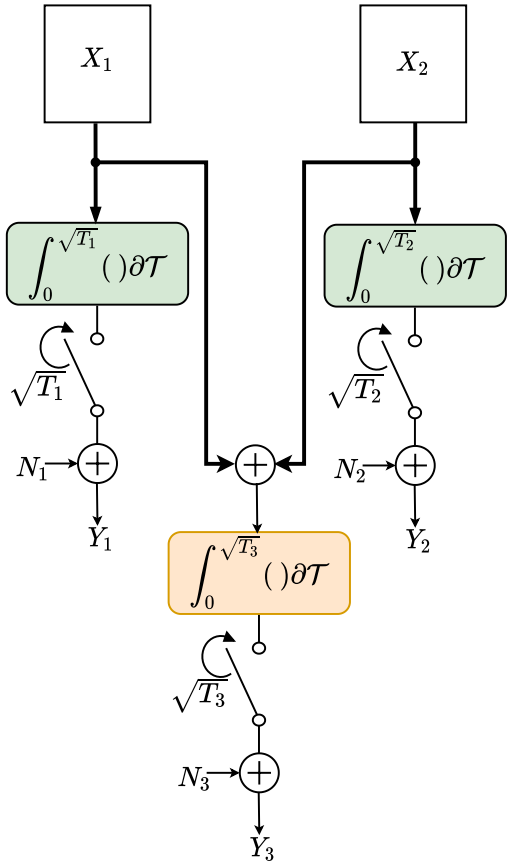

Figure 3.1: Sensing paradigm for detection of $2-$long hidden random vector X from $3-$long observable random vector Y through a vector Gaussian channel under a total time constraint of $T = \sum_{i=1}^{3} T_i$. Each Y_i is the sum of: output of integrate and dump circuit; and $N_i \sim \mathcal{N}(\mu = 0, \sigma^2 = 1)$. Each N_i is i.i.d Gaussian random variable. Each X_i is i.i.d such that $P(X_i = \lambda_0) = 1 - p$ and $P(X_i = \lambda_1) = p$.

have taken metrics of MI and Bayes risk to seek an optimal solution for our sensing problem which is a continuous-state and continuous-time output problem under the total time constraint. We have used the Monte Carlo method for our computations. It was observed through computational results that minimizing the Bayes risk under $0-1$ loss function and maximizing the mutual information does not necessarily result in the same optimal solution under the given total energy constraint. It was further

observed that for our problem it is the (hybrid sensing) combination of investing total time available into: sensing from the individual components of the vector input and the sensing from the sum of input-vector components that maximize both the Bayes probability of total correct detections and mutual information.

3.1 Introduction

We consider a vector Gaussian channel of fixed identity covariance matrix and binary input signalling as illustrated in fig. (3.1). A transformation (scaling matrix Φ) is performed on the vector input signal. The purpose of this chapter is two fold: to find the optimal scaling matrix under the total time constraint by: i) maximizing the mutual information between the input and output random vectors, ii) maximizing the probability of MAP detection, w.r.t scaling matrix when subjected to an energy constraint. It was found that the two metrics lead to different optimal solutions (computationally) and therefore we may say that this particular design problem might be a counter-example where detection theory provides a solution that is different from the information theory result as illustrated in fig. (3.2).

In [17] a Gaussian channel $Y|X \sim \mathcal{N}(\sqrt{T} \cdot X, 1)$ is considered and discovered that mutual information I is concave function in T for arbitrary input distribution. Whereas in [10] the Poisson channel $Y|X \sim \mathrm{Poiss}(T \cdot X)$ is investigated and found

a result similar to the Gaussian channel: $I(T)$ is concave in T for arbitrary input distribution. In chapter 2 it was observed that concavity of I is preserved under linear time constraint in a vector Poisson channel with a $2-$long binary input signalling and a $3-$long conditionally Poisson vector. It was further observed from a computational viewpoint that MAP detector was not necessarily reaching to the same optimal argument as that was given by mutual information. Here we construct an analogous model to that of Poisson channel such that at least the concavity of I remains intact for the Gaussian channel too. Compared to vector Poisson channel, the literature on vector Gaussian channel is comparatively richer and may help in providing some insight into the Poisson channel.

In past work [11] a generalization of Bregman divergence is developed to unify the vector Gaussian and Poisson channel models from the perspective of the gradient of mutual information; and mutual information is considered for signal recovery and classification with an energy constraint $\mathtt{Tr}(\Phi\Phi^{\intercal}) = 1$. MAP estimation is used for the classification purpose in [11] using Monte Carlo method to first approximate the gradient and then gradient descent is employed for the classification problem. It was noted in [11] that mutual information well served the classification problem. In this work we attempted to use the detection theory criterion for signal classification and then compared with the information theoretic solution. Another work [18] provides some results relevant to Gaussian channels about the concavity of I w.r.t squared singular values of the scaling matrix when certain conditions on the channel

covariance and precoder matrix are satisfied. It is found that for our problem I is concave in affine space defined by $(T_1, T_2, T_3) := (\frac{T-T_3}{2}, \frac{T-T_3}{2}, T_3)$ where $0 \leq T_3 \leq T$. For a Gaussian channel $Y|X \sim \mathcal{N}(\Phi X, \sigma_n^2 I)$ with input $X \sim \mathcal{N}(0, \Sigma)$ where Σ is full rank covariance matrix; then the two solutions from maximizing the mutual information and minimizing the mean-square error in scaling matrix under the power constraint $\mathtt{Tr}(\Phi\Phi^\intercal)$ leads to the same optimal solution which is a water-filling power allocation i-e concentrate more power resource to modes that provide higher snr [6]. Our problem is different in the input signalling, and we took the detection theory criterion instead of the estimation theory (MMSE) and then compared the optimal solution with one obtained from MI using Monte Carlo computational method.

The rest of the chapter is organized as follows: Section 3.2 introduces the problem description, explaining the vector Gaussian channel under consideration. Section 3.3 provides the *information theoretic* model, while Section 3.4 describes the *detection theoretic* model of the problem. Section 3.5 discusses the computed results from the previous two sections. Finally, Section 3.6 concludes the third chapter.

3.2 Problem Description

We consider the vector Gaussian channel:

$$\overbrace{\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \end{bmatrix}}^{Y} = \overbrace{\begin{bmatrix} \sqrt{T_1} & 0 \\ 0 & \sqrt{T_2} \\ \sqrt{T_3} & \sqrt{T_3} \end{bmatrix}}^{\Phi} \overbrace{\begin{bmatrix} X_1 \\ X_2 \end{bmatrix}}^{X} + \overbrace{\begin{bmatrix} N_1 \\ N_2 \\ N_3 \end{bmatrix}}^{W}, \tag{3.1}$$

with X_1 and X_2 be two independent and identical distributed (i.i.d) transformed Bernoulli random variables with p being the probability of occurrence of 1. We consider probability mass function f of discrete random vector $X \equiv [X_1, X_2]^\intercal$ as

$$f_X(x) = \begin{cases} p^2 & x = \begin{bmatrix} \lambda_1 & \lambda_1 \end{bmatrix}^\intercal \\ (1-p)^2 & x = \begin{bmatrix} \lambda_0 & \lambda_0 \end{bmatrix}^\intercal \\ p(1-p) & x = \begin{bmatrix} \lambda_0 & \lambda_1 \end{bmatrix}^\intercal \quad \text{or} \quad x = \begin{bmatrix} \lambda_1 & \lambda_0 \end{bmatrix}^\intercal \end{cases} \tag{3.2}$$

Noise vector $W = [N_1\, N_2\, N_3]^\intercal$ is a multivariate Gaussian with zero mean and identity covariance matrix; and independent of input X. The constraint on the scaling matrix is $T_1 + T_2 + T_3 = T$. The conditional distribution of vector Y given X is a multivariate

Gaussian:

$$Y\Big|(X = x_1x_2) \sim \mathcal{N}\left(\begin{bmatrix} \sqrt{T_1} \cdot x_1 \\ \sqrt{T_2} \cdot x_2 \\ \sqrt{T_3} \cdot (x_1 + x_2) \end{bmatrix}, I \right). \tag{3.3}$$

The objective is optimal time-allocation, (T_1, T_2, T_3), of total available time resource, T, that would maximize the *reward* i.e. either the mutual information or probability of total correct detections. Mathematically we may write

$$\max_{T_1,T_2,T_3} I(X_1, X_2; Y_1, Y_2, Y_3) \text{ s.t. } T_1 + T_2 + T_3 = T. \tag{3.4}$$

From the *detection theoretic* aspect we maximize the Bayesian probability of total correct detections, P_d, of hidden random vector X from observable random vector Y, as

$$\max_{T_1,T_2,T_3} P_d \quad \text{s.t.} \quad T_1 + T_2 + T_3 = T \tag{3.5}$$

3.3 Information Theoretic Description

3.3.1 Scalar Gaussian channel

The scalar version of the Gaussian channel is first presented, and then we extend it to the vector version. We start with mutual information between a scalar random variable X_1 which is a scaled Bernoulli random variable and Y_1 is a univariate Gaussian mixture. The probability density function of Y_1 is then given as

$$f(Y_1) = (1-p)\ \cdot \mathcal{N}(Y_1; \lambda_0\sqrt{T}, 1) + p\ \cdot \mathcal{N}(Y_1; \lambda_1\sqrt{T}, 1).$$

The mutual information I can be written as

$$I(X_1; Y_1) = H(Y_1) - H(Y_1|X_1)$$

where $H(\cdot)$ is the Shannon entropy and f is the probability mass function of random variate Y with $\mathcal{Y}$ as the corresponding support. We may write differential entropy

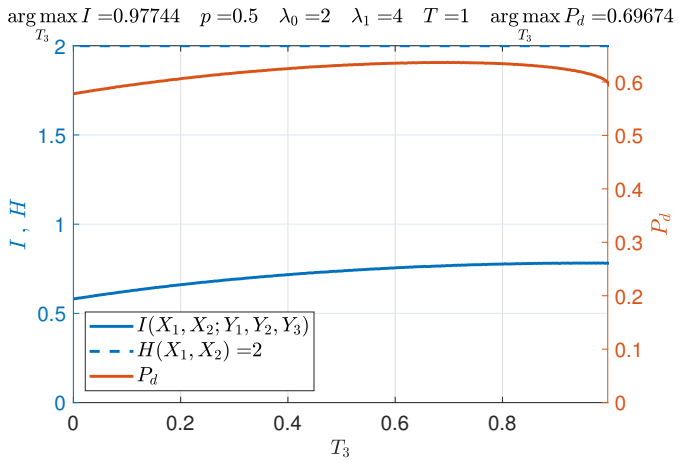

Figure 3.2: Mutual information $I(X;Y)$ vs. T_3 and probability of total correct detections Pd vs. time T_3 in a time constraint $T_1 + T_2 + T_3 = 1$ such that $(T_1, T_2, T_3) := (\frac{1-T_3}{2}, \frac{1-T_3}{2}, T_3)$ where $0 \leq T_3 \leq 1$.
(MATLAB-CODE F.9 & F.18.4)

$H(Y_1)$ as

$$H(Y_1) = -\int_{-\infty}^{\infty} \Big((1-p) \cdot \mathcal{N}(y_1; \lambda_0\sqrt{T}, 1) + p \cdot \mathcal{N}(y_1; \lambda_1\sqrt{T}, 1) \cdot \\ \mathrm{Log}_2 \Big[(1-p) \cdot \mathcal{N}(y_1; \lambda_0\sqrt{T}, 1) + p \cdot \mathcal{N}(y_1; \lambda_1\sqrt{T}, 1) \Big] \Big) \, dy_1, \tag{3.6}$$

and

$$H(Y_1|X_1) = (1-p) \cdot 0.5 \cdot \mathrm{Log}_2[2\pi\, e] + p \cdot 0.5 \cdot \mathrm{Log}_2[2\pi\, e]. \tag{3.7}$$

It is noted in [17] that mutual information given above is a concave function in T. In the following section we formulate the mutual information expression for our vector Gaussian model.

3.3.2 Vector Gaussian channel

Mutual information between two random vectors can be defined as the difference between the total differential entropy in one random vector and the conditional differential entropy in the second random vector given the first vector. We write

$$I(X;Y) = H(Y) - H(Y|X) \tag{3.8}$$

The conditional entropy $H(Y|X)$ is calculated from the conditional probability mass functions $f(Y|X_i)$ defined as

$$
\begin{aligned}
& f(Y|X = [\lambda_0 \quad \lambda_0]^\intercal) \\
&= \mathcal{N}(Y_1; \lambda_0\sqrt{T_1}, 1) \cdot \mathcal{N}(Y_2; \lambda_0\sqrt{T_2}, 1) \cdot \mathcal{N}(Y_3; 2\lambda_0\sqrt{T_3}, 1), \\
& f(Y|X = [\lambda_0 \quad \lambda_1]^\intercal) \\
&= \mathcal{N}(Y_1; \lambda_0\sqrt{T_1}, 1) \cdot \mathcal{N}(Y_2; \lambda_1\sqrt{T_2}, 1) \cdot \mathcal{N}(Y_3; (\lambda_0 + \lambda_1)\sqrt{T_3}, 1), \\
& f(Y|X = [\lambda_1 \quad \lambda_0]^\intercal) \\
&= \mathcal{N}(Y_1; \lambda_1\sqrt{T_1}, 1) \cdot \mathcal{N}(Y_2; \lambda_0\sqrt{T_2}, 1) \cdot \mathcal{N}(Y_3; (\lambda_1 + \lambda_0)\sqrt{T_3}, 1), \\
& f(Y|X = [\lambda_1 \quad \lambda_1]^\intercal) \\
&= \mathcal{N}(Y_1; \lambda_1\sqrt{T_1}, 1) \cdot \mathcal{N}(Y_2; \lambda_1\sqrt{T_2}, 1) \cdot \mathcal{N}(Y_3; 2\lambda_1\sqrt{T_3}, 1).
\end{aligned}
$$

The marginal probability mass function of Y is then given as

$$
\begin{aligned}
& f(Y) \\
&= (1-p)^2 \cdot f(Y|X = [\lambda_0 \quad \lambda_0]^\intercal) + p(1-p)\cdot \\
& f(Y|X = [\lambda_0 \quad \lambda_1]^\intercal) + p(1-p) \cdot f(Y|X = [\lambda_1 \quad \lambda_0]^\intercal) + \\
& p^2 \cdot f(Y|X = [\lambda_1 \quad \lambda_1]^\intercal).
\end{aligned} \tag{3.9}
$$

Mutual information $I(X;Y)$ is then defined as

$$
I(X;Y) = H(Y) - H(Y|X), \tag{3.10}
$$

where $H(Y)$ is a differential entropy of our finite Gaussian mixture model (gmm) and given as

$$
\begin{aligned}
H(Y) = -\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \Bigg[\Bigg(& (1-p)^2 \cdot \mathcal{N}(y_1;\lambda_0 T_1,1)\cdot\mathcal{N}(y_2;\lambda_0 T_2,1)\cdot\mathcal{N}(y_3;2\lambda_0 T_3,1)+ \\
& p(1-p)\cdot\mathcal{N}(y_1;\lambda_0 T_1,1)\cdot\mathcal{N}(y_2;\lambda_1 T_2,1)\cdot\mathcal{N}(y_3;(\lambda_0+\lambda_1) T_3,1)+ \\
& p(1-p)\cdot\mathcal{N}(y_1;\lambda_1 T_1,1)\cdot\mathcal{N}(y_2;\lambda_0 T_2,1)\cdot\mathcal{N}(y_3;(\lambda_1+\lambda_0) T_3,1)+ \\
& p^2\cdot\mathcal{N}(y_1;\lambda_1 T_1,1)\cdot\mathcal{N}(y_2;\lambda_1 T_2,1)\cdot\mathcal{N}(y_3;2\lambda_1 T_3,1)\Bigg)\cdot \\
& \Bigg(\mathrm{Log}_2[(1-p)^2\cdot\mathcal{N}(y_1;\lambda_0 T_1,1)\cdot\mathcal{N}(y_2;\lambda_0 T_2,1)\cdot\mathcal{N}(y_3;2\lambda_0 T_3,1)+ \\
& p(1-p)\cdot\mathcal{N}(y_1;\lambda_0 T_1,1)\cdot\mathcal{N}(y_2;\lambda_1 T_2,1)\cdot\mathcal{N}(y_3;(\lambda_0+\lambda_1) T_3,1)+ \\
& p(1-p)\cdot\mathcal{N}(y_1;\lambda_1 T_1,1)\cdot\mathcal{N}(y_2;\lambda_0 T_2,1)\cdot\mathcal{N}(y_3;(\lambda_1+\lambda_0) T_3,1)+ \\
& p^2\cdot\mathcal{N}(y_1;\lambda_1 T_1,1)\cdot\mathcal{N}(y_2;\lambda_1 T_2,1)\cdot\mathcal{N}(y_3;2\lambda_1 T_3,1)]\Bigg)\Bigg]\, dy_1\, dy_2\, dy_3, \qquad (3.11)
\end{aligned}
$$

and $H(Y|X)$ is

$$
\begin{aligned}
H(Y|X) = & (1-p)^2\cdot(0.5\cdot\mathrm{Log}_2[2\pi\, e]+0.5\cdot\mathrm{Log}_2[2\pi\, e]+0.5\cdot\mathrm{Log}_2[2\pi\, e])+ \\
& p(1-p)\cdot(0.5\cdot\mathrm{Log}_2[2\pi\, e]+0.5\cdot\mathrm{Log}_2[2\pi\, e]+0.5\cdot\mathrm{Log}_2[2\pi\, e)+ \\
& p(1-p)\cdot(0.5\cdot\mathrm{Log}_2[2\pi\, e]+0.5\cdot\mathrm{Log}_2[2\pi\, e]+0.5\cdot\mathrm{Log}_2[2\pi\, e]) \\
& p^2\cdot(0.5\cdot\mathrm{Log}_2[2\pi\, e]+0.5\cdot\mathrm{Log}_2[2\pi\, e]+0.5\cdot\mathrm{Log}_2[2\pi\, e]). \qquad (3.12)
\end{aligned}
$$

Since the multidimensional integral defined in (3.11) has no closed-form solution, we have to resort to numerical methods. We may mitigate the curse of dimensionality involved in multi-dimensional integration by Monte-Carlo technique by

taking samples from the multivariate Gaussian mixture distribution to achieve fast convergence to the true mixture differential entropy at a reasonable computational burden; whereas naive uniform sampling of the space would lead to a quite slow convergence to the true differential entropy.

$$\begin{aligned} H(Y) &= E[-\operatorname{Log}_2[f_Y(Y)]] \\ &= -\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f_Y(y)\cdot\operatorname{Log}_2[f_Y(y)]\,dy \\ &\approx -\frac{\sum_i \operatorname{Log}_2[f_Y(s_i)]}{N_s}, \end{aligned} \tag{3.13}$$

where $f_Y(\cdot)$ is the mixture probability distribution of Y, N_s is the number of MC samples and s_i is the i^{th} sample from multivariate Gaussian mixture distribution. Fig.(3.2) illustrates the concavity of mutual information in T_3 of a vector Gaussian channel when time constraint is imposed.

Theorem 3.1 *$I(X_1, X_2; Y_1, Y_2, Y_3)$ is symmetric in variables T_1 and T_2.*

Proof: Mutual information $I(X_1, X_2; Y_1, Y_2, Y_3)$ given in (3.10) is invariant under any permutation of variables T_1 and T_2. That means interchanging the two variables leaves the expression unchanged. ■

Theorem 3.2 *$I(X_1, X_2; Y_1, Y_2, Y_3)$ is concave in $T_3 = 0$ plane.*

Proof:

$I(X_1, X_2; Y_1, Y_2, Y_3)\Big|_{T3=0} = I(X_1, X_2; Y_1, Y_2)$

By chain rule of mutual information:

$$\begin{aligned} I(X_1, X_2; Y_1, Y_2) &= I(X_1, X_2; Y_1) + I(X_1, X_2; Y_2|Y_1) \\ &= I(X_1; Y_1) + I(X_2; Y_2). \end{aligned} \tag{3.14}$$

We note in (3.14) that each $I(X_i; Y_i)$ is solely a function of T_i and also concave in it [17]. Since the sum of concave functions is a concave function. Therefore $I(X_1, X_2; Y_1, Y_2)$ is concave in T_1 and T_2 when $T_3 = 0$. This concludes the proof. ■

Theorem 3.3 *$I(X_1, X_2; Y_1, Y_2, Y_3)$ is concave in T_3 along the line $(T_1, T_2, T_3) := (\frac{T-T_3}{2}, \frac{T-T_3}{2}, T_3)$ parameterized by $0 \leq T_3 \leq T$.*

Proof: It is noted in [7, Theorem 5] that mutual information is a concave function of the squared singular values ($\boldsymbol{\lambda}$) of the precoder matrix P if the first $m^{'}$ eigenvectors of the channel covariance matrix ($\mathbf{R}_H = H^{\intercal}\mathbf{R}_Z^{-1}H$) coincide with the left singular vectors of the precoder P i-e $\mathsf{H}_{\lambda} I(S; Y) \leq 0$ for the signal model $Y = HPS + Z$ where $H \in \mathbb{R}^{n \times p}$ is the channel, S is the input signaling $S \in \mathbb{R}^m$, P is a precoder matrix $P \in \mathbb{R}^{p \times m}$ and $Z \in \mathbb{R}^n$ is Gaussian noise independent of the input S and has covariance matrix $\mathbf{R}_Z$.

For our problem: $H = I$, $\mathbf{R}_Z^{-1} = \Lambda = I$, $P = \Phi$, $S = X$ and $Z = W$. The singular value decomposition of $\Phi = U\Sigma V^*$. We have singular matrix

$$\Sigma = \begin{bmatrix} \frac{\sqrt{T_1+T_2+2T_3-\sqrt{(T_1-T_2)^2+4T_3^2}}}{\sqrt{2}} & 0 \\ 0 & \frac{\sqrt{T_1+T_2+2T_3+\sqrt{(T_1-T_2)^2+4T_3^2}}}{\sqrt{2}} \\ 0 & 0 \end{bmatrix} \tag{3.15}$$

By substituting $T_1 = T_2 = \frac{T-T_3}{2}$, the squared singular values are $[\lambda_1 \quad \lambda_2 \quad \lambda_3] = [\frac{T-T_3}{2} \quad \frac{T+3\cdot T_3}{2} \quad 0]$ for $0 \leq T_3 \leq T$. This is just the composition with an affine transformation on the domain. Concavity remains preserved under affine transformation [12, page 79-86]. ■

3.4 Detection Theoretic Description

3.4.1 Bayes risk

In last section, we presented the metric of mutual information I between hidden random vector X, and observable vector Y. Here we approach the sensing problem as a multi-hypothesis detection problem and define the Bayesian risk [16, pp.220]

to minimize in (T_1, T_2, T_3). We define Bayes risk r as

$$
\begin{aligned}
r = (1-p)^2 \Big[& P_{\lambda_0\lambda_0 \mid \lambda_0\lambda_0}\, C_{\lambda_0\lambda_0 \mid \lambda_0\lambda_0} + P_{\lambda_0\lambda_1 \mid \lambda_0\lambda_0}\, C_{\lambda_0\lambda_1 \mid \lambda_0\lambda_0} \\
& + P_{\lambda_1\lambda_0 \mid \lambda_0\lambda_0}\, C_{\lambda_1\lambda_0 \mid \lambda_0\lambda_0} + P_{\lambda_1\lambda_1 \mid \lambda_0\lambda_0}\, C_{\lambda_1\lambda_1 \mid \lambda_0\lambda_0} \Big] + p(1-p) \\
& \Big[P_{\lambda_0\lambda_0 \mid \lambda_0\lambda_1}\, C_{\lambda_0\lambda_0 \mid \lambda_0\lambda_1} + P_{\lambda_0\lambda_1 \mid \lambda_0\lambda_1}\, C_{\lambda_0\lambda_1 \mid \lambda_0\lambda_1} + P_{\lambda_1\lambda_0 \mid \lambda_0\lambda_1}\, C_{\lambda_1\lambda_0 \mid \lambda_0\lambda_1} \\
& + P_{\lambda_1\lambda_1 \mid \lambda_0\lambda_1}\, C_{\lambda_1\lambda_1 \mid \lambda_0\lambda_1} \Big] + p(1-p) \Big[P_{\lambda_0\lambda_0 \mid \lambda_1\lambda_0}\, C_{\lambda_0\lambda_0 \mid \lambda_1\lambda_0} \\
& + P_{\lambda_0\lambda_1 \mid \lambda_1\lambda_0}\, C_{\lambda_0\lambda_1 \mid \lambda_1\lambda_0} + P_{\lambda_1\lambda_0 \mid \lambda_1\lambda_0}\, C_{\lambda_1\lambda_0 \mid \lambda_1\lambda_0} + P_{\lambda_1\lambda_1 \mid \lambda_1\lambda_0}\, C_{\lambda_1\lambda_1 \mid \lambda_1\lambda_0} \Big] \\
& + p^2 \Big[P_{\lambda_0\lambda_0 \mid \lambda_1\lambda_1}\, C_{\lambda_0\lambda_0 \mid \lambda_1\lambda_1} + P_{\lambda_0\lambda_1 \mid \lambda_1\lambda_1}\, C_{\lambda_0\lambda_1 \mid \lambda_1\lambda_1} + P_{\lambda_1\lambda_0 \mid \lambda_1\lambda_1}\, C_{\lambda_1\lambda_0 \mid \lambda_1\lambda_1} \\
& + P_{\lambda_1\lambda_1 \mid \lambda_1\lambda_1}\, C_{\lambda_1\lambda_1 \mid \lambda_1\lambda_1} \Big],
\end{aligned}
$$

where $P_{\lambda_k\lambda_l \mid \lambda_i\lambda_j}$ is the probability that $X = [\lambda_i, \lambda_j]^\intercal$ is true while decision $X = [\lambda_k, \lambda_l]^\intercal$ is made; similarly for $C_{\lambda_k\lambda_l \mid \lambda_i\lambda_j}$. Setting all costs for which $[\lambda_i, \lambda_j]^\intercal \neq [\lambda_k, \lambda_l]^\intercal$ to one and $[\lambda_i, \lambda_j]^\intercal = [\lambda_k, \lambda_l]^\intercal$ to zero, we have

$$
\begin{aligned}
r = (1-p)^2 & \Big[P_{\lambda_0\lambda_1 \mid \lambda_0\lambda_0} + P_{\lambda_1\lambda_0 \mid \lambda_0\lambda_0} + P_{\lambda_1\lambda_1 \mid \lambda_0\lambda_0} \Big] + p(1-p) \\
& \Big[P_{\lambda_0\lambda_0 \mid \lambda_0\lambda_1} + P_{\lambda_1\lambda_0 \mid \lambda_0\lambda_1} + P_{\lambda_1\lambda_1 \mid \lambda_0\lambda_1} \Big] + p(1-p) \\
& \Big[P_{\lambda_0\lambda_0 \mid \lambda_1\lambda_0} + P_{\lambda_0\lambda_1 \mid \lambda_1\lambda_0} + P_{\lambda_1\lambda_1 \mid \lambda_1\lambda_0} \Big] + p^2 \\
& \Big[P_{\lambda_0\lambda_0 \mid \lambda_1\lambda_1} + P_{\lambda_0\lambda_1 \mid \lambda_1\lambda_1} + P_{\lambda_1\lambda_0 \mid \lambda_1\lambda_1} \Big].
\end{aligned} \tag{3.16}
$$

We are interested in minimizing this Bayes risk r in (T_1, T_2, T_3) i-e

$$\min_{T_1,T_2,T_3} r \quad \text{s.t. } T_1 + T_2 + T_3 = T. \tag{3.17}$$

Note that while minimizing r in (T_1, T_2, T_3), the decision boundaries would be changing accordingly and become function of (T_1, T_2, T_3). Equivalently, we may say that

$$\max_{T_1,T_2,T_3} P_d \quad \text{s.t. } T_1 + T_2 + T_3 = T \tag{3.18}$$

where P_d is probability of total correct detections, $P_d = 1 - r$. In the next section we present the computed results of (3.18).

3.5 Monte Carlo Simulation Results

For all simulation purposes, we have assumed that optimizing argument in $\max_{T_1,T_2,T_3}$ $I(X_1, X_2; Y_1, Y_2, Y_3)$ s.t. $T_1 + T_2 + T_3 = T$ would have $T_1 = T_2$. This is based on the observations noted in the ternary diagrams given in fig.(3.3). We computed I for a wide range of given input parameters λ_0, λ_1, T and p; and it was noted that the maximizing argument always seems to lie on the line $(T_1, T_2, T_3) := (\frac{T-\alpha}{2}, \frac{T-\alpha}{2}, \alpha)$ where $0 \leq \alpha \leq T$. In other words we noted a Schur concavity of I in (T_1, T_2)

whenever T_3 is held fixed under a given time-constraint; however no proof of Schur concavity of I is claimed in this work.

We first compute mutual information in (3.11) by generating the samples from the multivariate Gaussian mixture distribution. Each of the Gaussian mixture component is a $3-$dimensional multivariate Gaussian distribution that comes with a prior belief. We generate a total of 10^6 samples to calculate $H(Y)$ for a given prior p and energy constraint $T_1+T_2+T_3 = T$ with $T_1 = T_2$. Since we do have a closed-form available for a differential entropy of a multivariate Gaussian distribution therefore for $H(Y|X)$ we do not need to apply the Monte Carlo method for evaluating it. The difference of the two would provide the approximated value of $I(X;Y)\Big|_{(\frac{T-\alpha}{2},\frac{T-\alpha}{2},\alpha)}$ for a given set of parameters.

For the MAP detection we use the empirical method to calculate the probability of total correct detections P_d. We again assume that the optimal solution has $T_1 = T_2$. An optimal solution for any given set of parameters is then searched in the region $(T_1, T_2, T_3) := (\frac{T-\alpha}{2}, \frac{T-\alpha}{2}, \alpha)$ where $0 \leq \alpha \leq T$. For a given value of T: α takes 400 linear steps from 0 to T and for each step we first generate the samples from the Gaussian mixture under consideration by additionally knowing which mixture component has actually generated any particular sample. For every input sample we then computed the posterior probability for each of four hypotheses and then decide in favor of the hypothesis that has the highest posterior probability. Comparing our

10^6 decisions with that of the 10^6 inputs, we can then calculate the total correct detections for each of the discrete α steps. This way for any given set of parameters λ_0, λ_1, T, p and α we may empirically compute the $P_d\Big|_{(\frac{T-\alpha}{2},\frac{T-\alpha}{2},\alpha)}$.

For the sake of simplicity we call the time proportion: $(\frac{T}{2},\frac{T}{2},0)$ to be the individual sensing; $(0,0,T)$ to be the joint sensing and $(\frac{T-\alpha}{2},\frac{T-\alpha}{2},\alpha)$ where $0<\alpha<T$ to be hybrid sensing method. In fig.(3.4) and fig.(3.5), there are a couple of observations to be noted: first we can see the concavity of I and P_d in T_3; second observation is maximizing the mutual information and probability of total correct detections does not lead to the same optimal solution; this is more noticeable in fig.(3.5) where mutual information is maximized in the vicinity of $T_3=1$ and therefore suggesting individual sensing to be optimum whereas probability of total correct detections is suggesting the hybrid sensing to be optimum. The third observation is that just by looking at the prior p we can not say in the most rough sense that which of the three sensing mechanisms would be optimal, either from the perspective of the mutual information or from the Bayes inference. This is unlike to that of a Poisson problem discussed in chapter 2 where individual sensing was always optimal whenever $p\geq 0.5$ irrespective of the given input set of parameters from mutual information perspective.

To further expand our understanding if hybrid sensing remains optimal for a wide range of input parameters λ_0 and λ_1 for fixed prior p and time constraint $T=1$,

we perform another Monte Carlo simulation. The input parameter set is $\{(\lambda_0, \lambda_1) \in \mathbb{R}_+ \times \mathbb{R}_+ \big| 0 < \lambda_1 \leq 5 \text{ and } \lambda_1 > \lambda_0\}$. For each of (λ_0, λ_1) we compute 400 values of mutual information by varying α linearly from 0 to $T = 1$ in 400 steps. For each step 10^5 samples are used for calculation of differential entropy $H(Y)$. Scatter plots on the left-hand side in fig.(3.6) illustrates the respective optimal value of $I(X;Y)$ at each input parameter for the prior taking values: $0.125, 0.5$ and 0.99. Whereas the scatter plots on the right-hand side illustrates the corresponding optimizing argument $(\frac{T-T_3}{2}, \frac{T-T_3}{2}, T_3)$ where $0 < T_3 < T$. It can be seen that when the two input parameters λ_0 and λ_1 are closer (as in the diagonal) the mutual information is near to zero and hybrid sensing is the best sensing strategy; as the two input parameters gets farther (as in the lower right corner in scatter plot) the mutual information gets higher and still the hybrid sensing is optimal. This is true for all three values of the prior. When the same simulation is run for maximizing the Bayes probability of total correct detections P_d the results are shown in scatter plots of fig.(3.7). P_d is shown on the left scatter plot for each prior. As the input parameters λ_0 and λ_1 gets closer (as in the diagonal) the P_d touches the maximum value among the $\{(1-p)^2, \quad 2p(1-p), \quad p^2\}$. In the lower right corner the P_d is highest as the input parameters are the farthest apart. The right scatter plots illustrates that hybrid sensing is the optimum method from the Bayes detection point of view. It must be noted that even the hybrid sensing is optimal from perspectives of the mutual information and Bayes probability of total correct detection; the optimal arguments

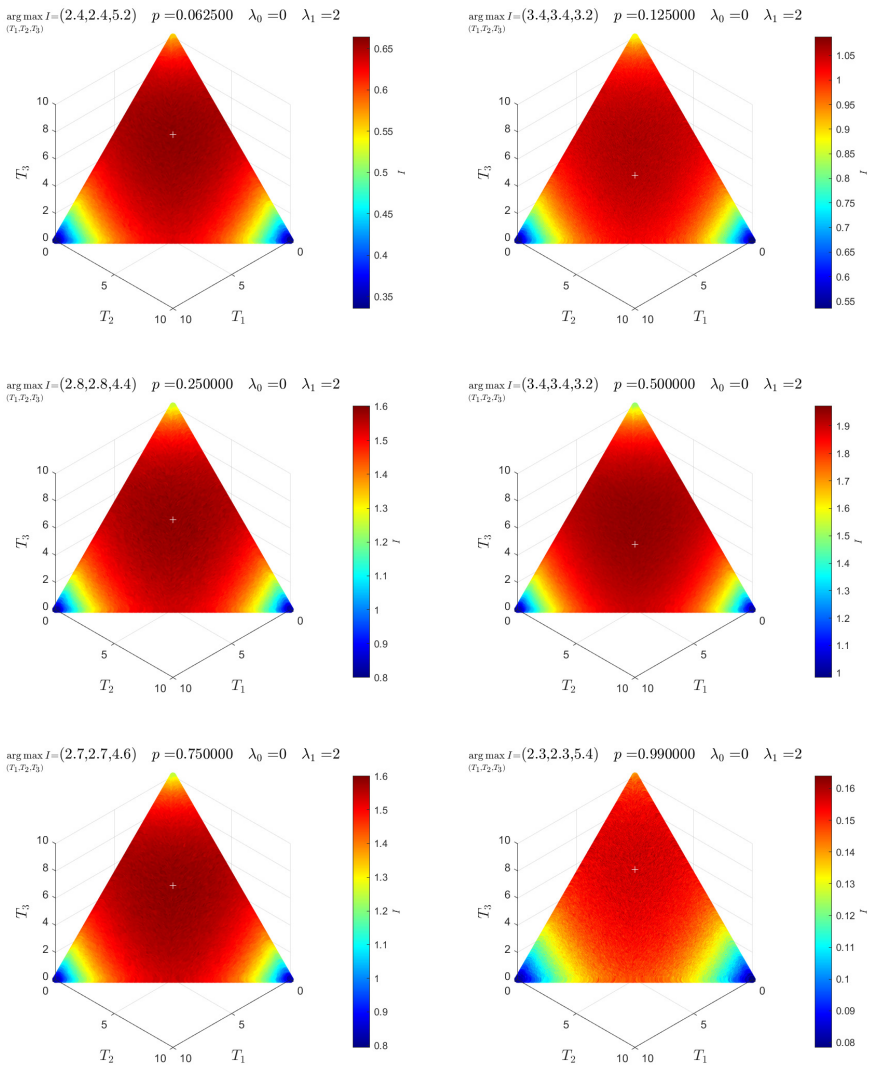

Figure 3.3: $I(X;Y)$ vs. (T_1, T_2, T_3) under time constraint $T_1 + T_2 + T_3 = 10$ for $\lambda_0 = 0$, $\lambda_1 = 2$, and varying *prior* probability p.
(MATLAB-CODE F.10 & F.18.5)

from these two metrics are not necessarily appear to be the same. These simulations therefore constitute a counter-example where information theory and detection theory are leading to different optimal solutions.

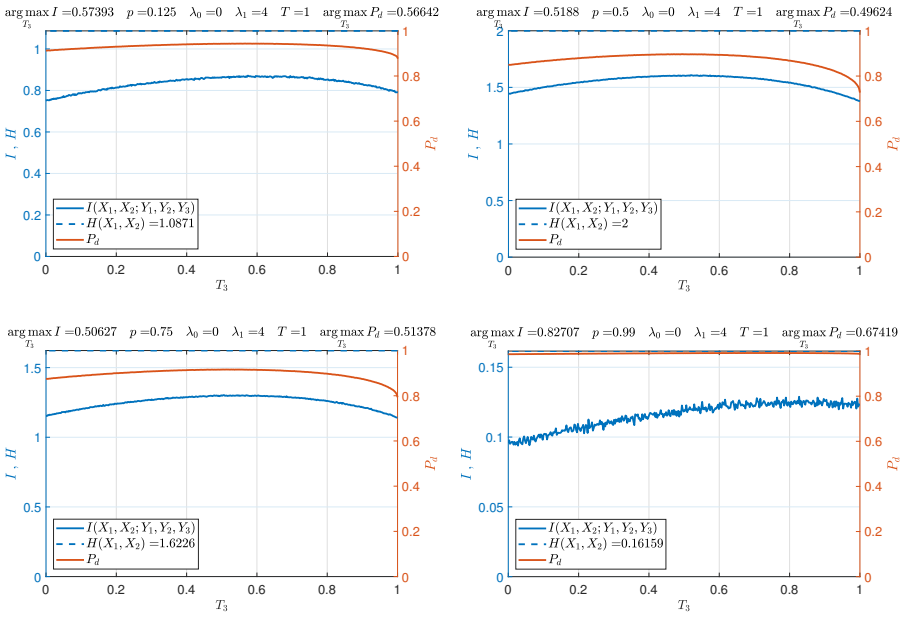

Figure 3.4: Mutual information $I(X;Y)$ vs. T_3 and probability of total correct detections P_d vs. T_3 for *prior* probabilities of $0.125, 0.5, 0.75$ and 0.99. (MATLAB-CODE F.11 & F.18.4)

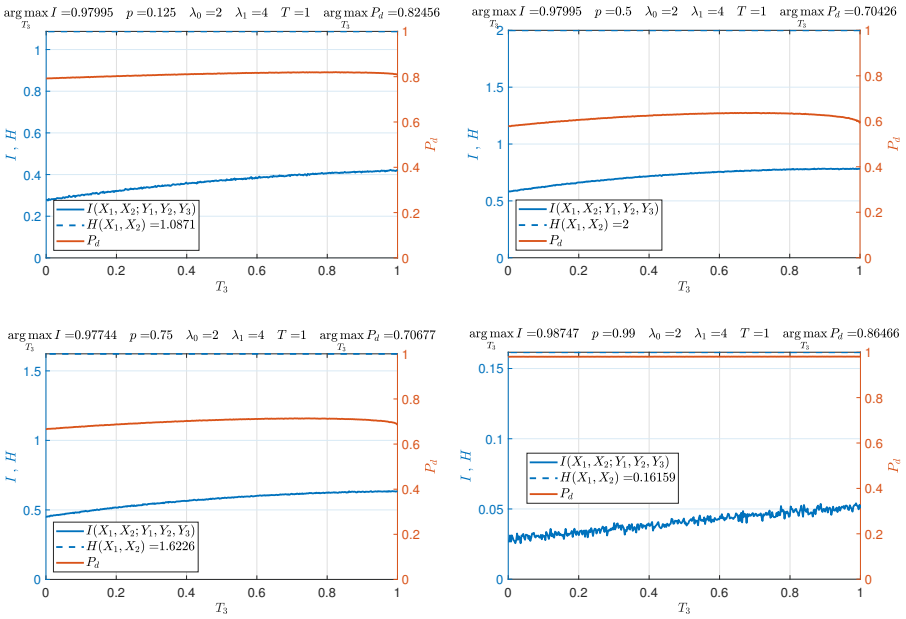

Figure 3.5: Mutual information $I(X;Y)$ vs. T_3 and probability of total correct detections P_d vs. T_3 for *prior* probabilities of $0.125, 0.5, 0.75$ and 0.99. (MATLAB-CODE F.11 & F.18.4)

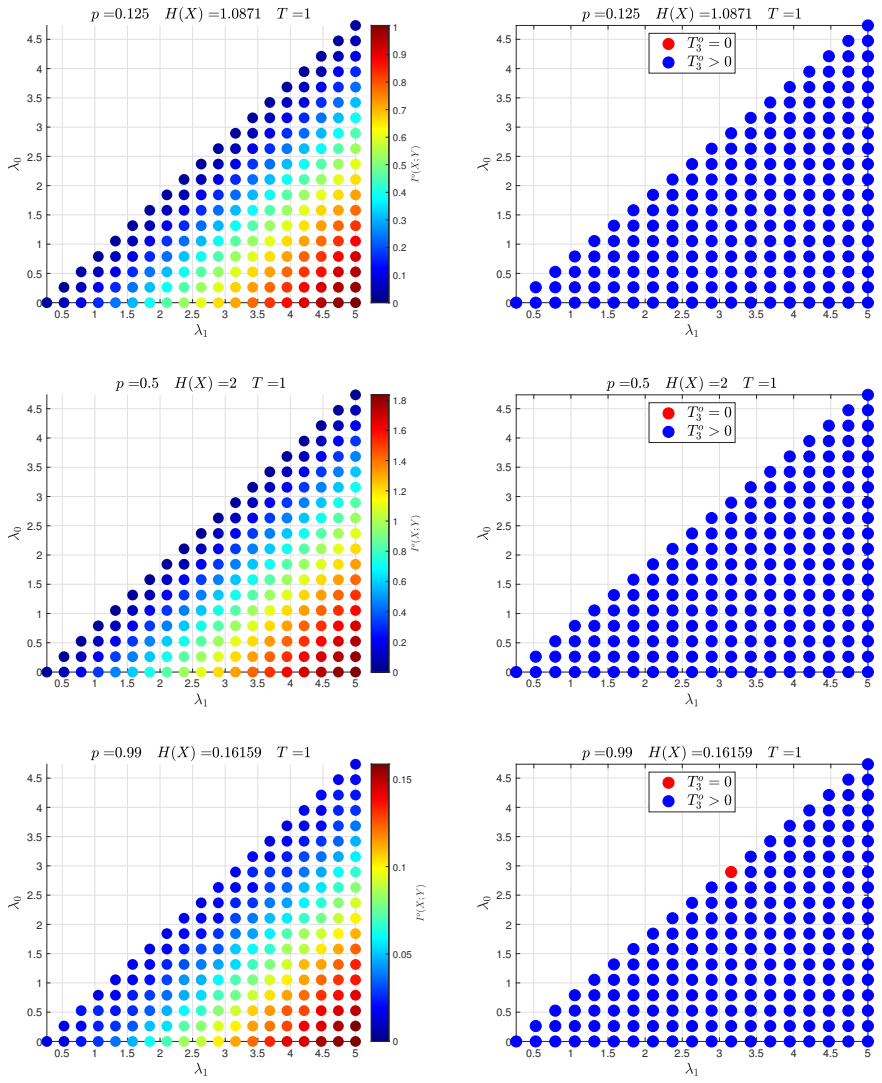

Figure 3.6: Left: $I(X;Y)^O$ vs. (λ_0, λ_1) in the region $\lambda_1 > \lambda_0$, right: corresponding optimal argument parameter T_3^o vs. (λ_0, λ_1) for varying *prior* probabilities p. The search for each optimal argument T_3^o for any fixed: (λ_0, λ_1) and p is performed over the line $(T_1, T_2, T_3) := (\frac{1-T_3}{2}, \frac{1-T_3}{2}, T_3)$ where $0 \leq T_3 \leq 1$. (MATLAB-CODE F.12 & F.18.4)

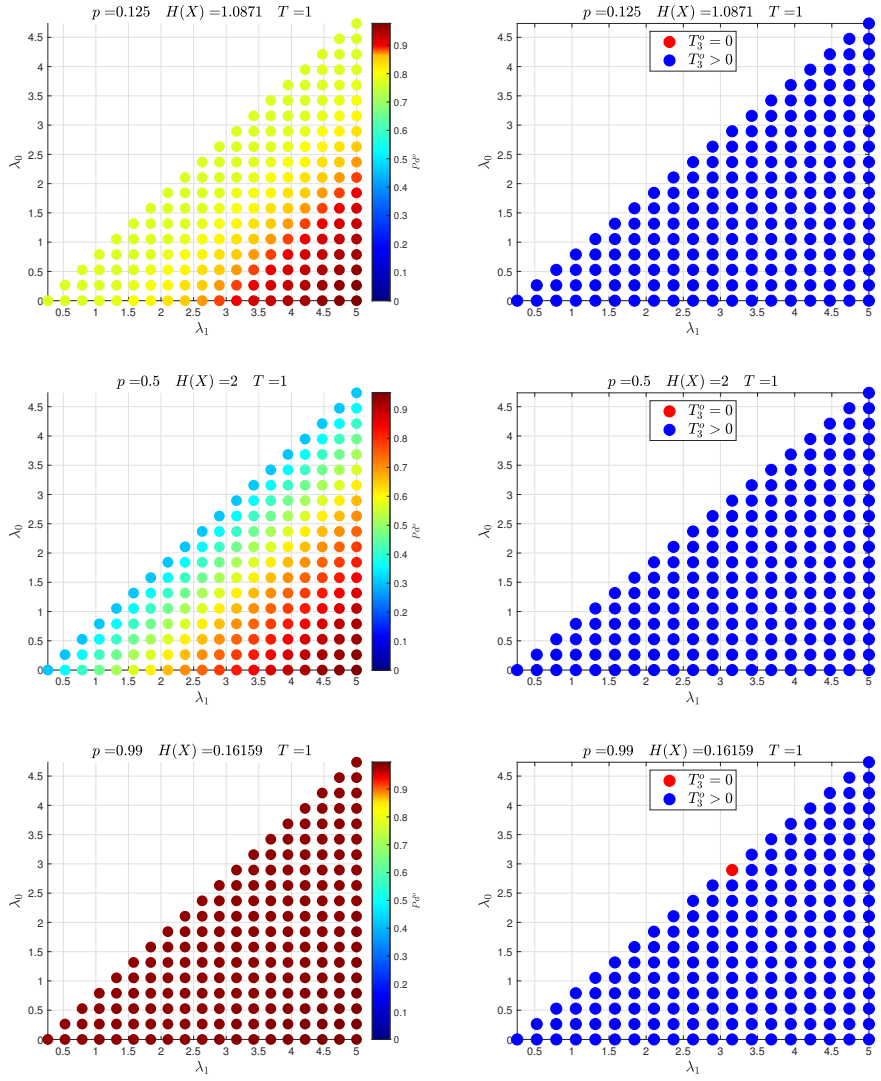

Figure 3.7: Left: P_d^O vs. (λ_0, λ_1) in the region $\lambda_1 > \lambda_0$, right: corresponding optimal argument parameter T_3^o vs. (λ_0, λ_1) for varying *prior* probabilities p. The search for each optimal argument T_3^o for any fixed: (λ_0, λ_1) and p is performed over the line $(T_1, T_2, T_3) := (\frac{1-T_3}{2}, \frac{1-T_3}{2}, T_3)$ where $0 \leq T_3 \leq 1$. (MATLAB-CODE F.12 & F.18.4)

3.6 Conclusion

This work attempts to address the problem of sensor scheduling in a vector Gaussian channel for a two target detection, when a specified structure on scaling matrix is imposed, using criteria of mutual information and Bayesian risk with $0-1$ loss function. From computations, it was found that the optimal argument under mutual information criterion need not necessarily be optimal under Bayesian inference. It was found that both mutual information and Bayes probability of total correct detections is concave in α in the line $(T_1, T_2, T_3) := (\frac{T-\alpha}{2}, \frac{T-\alpha}{2}, \alpha)$ parameterized by $0 \leq \alpha \leq T$. The scaling matrix that we considered has shown (computationally) the concave nature of mutual information in affine spaces of interest under linear time constraint. For any given prior p and given finite time T, hybrid sensing is found to be the optimal sensing mechanism for any given time proportions.

It would be interesting to know some analytical evidence of the concavity or non-concavity of mutual information in the line $(T_1, T_2, T_3) := (\alpha \cdot T, (1-\alpha) \cdot T, c)$ parameterized by $0 \leq \alpha \leq 1$ for some positive constant c.

Chapter 4

Heuristic Sensing Schemes for Four-Target Detection in Time-Constrained Vector Poisson and Gaussian Channels

In this chapter we investigate the different sensing schemes for detection of four targets when observed through a vector Poisson and Gaussian channels. For this purpose mutual information and Bayes risk are used to maximize the information and detection respectively, for any given fixed time. It is observed that for both Poisson and Gaussian channels; mutual information and Bayes risk with $0-1$ cost

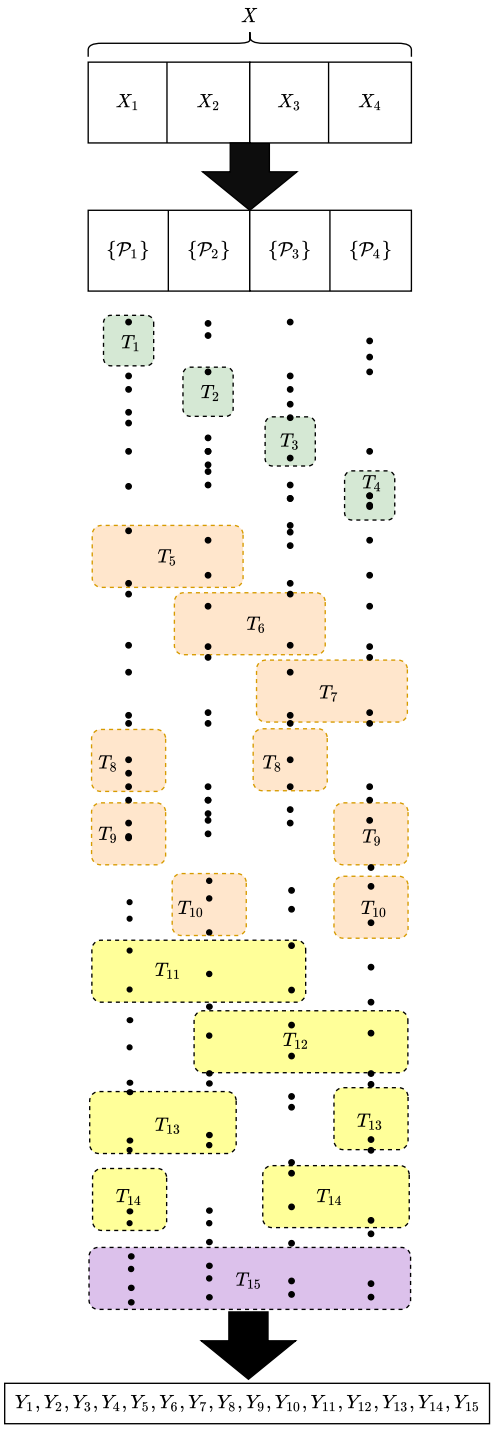

Figure 4.1: Illustration of sensing paradigm for detection of $4-$long hidden random vector X from $15-$long observable random vector Y through a vector Poisson channel under a total time constraint of $T = \sum_{i=1}^{15} T_i$. Each X_i is i.i.d and $P(X_i = \lambda_0) = 1 - p$ and $P(X_i = \lambda_1) = p$.

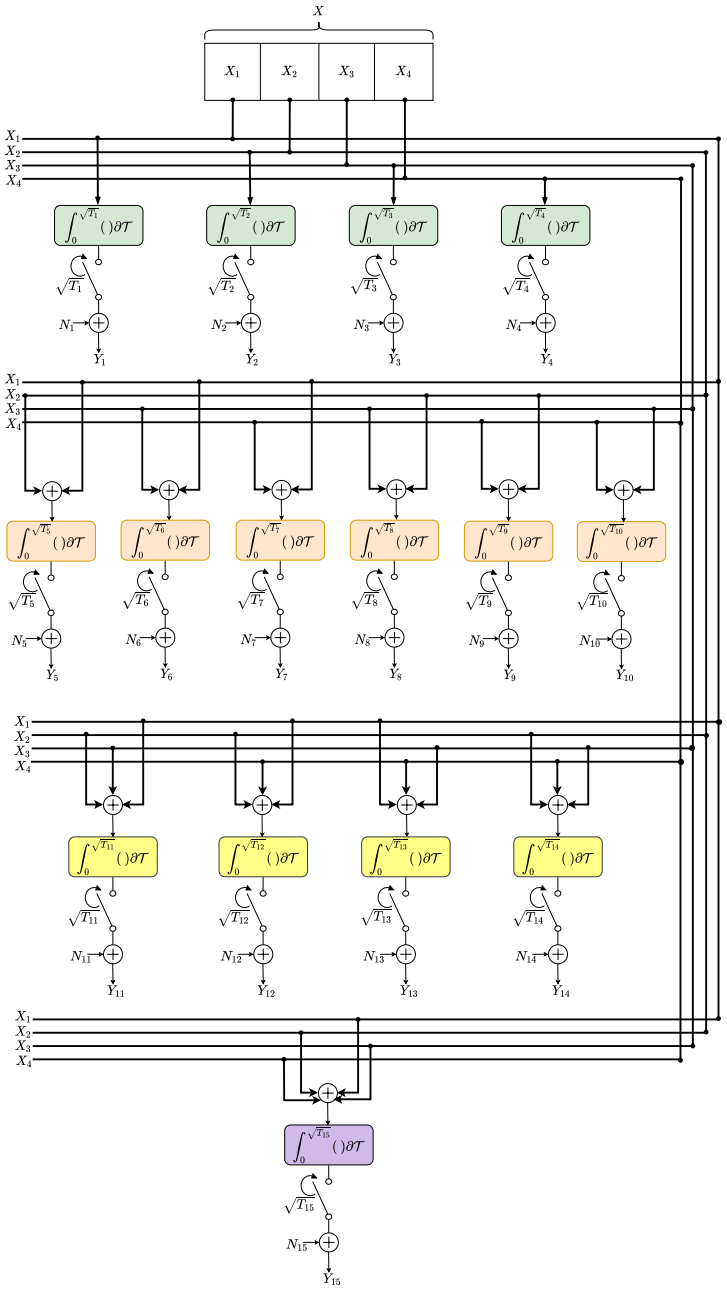

Figure 4.2: Sensing paradigm for detection of $4-$long hidden random vector X from $15-$long observable random vector Y through a vector Gaussian channel under a total time constraint of $T = \sum_{i=1}^{15} T_i$. Each Y_i is the sum of: output of integrate and dump circuit; and $N_i \sim \mathcal{N}(\mu = 0, \sigma^2 = 1)$. Each N_i is i.i.d Gaussian random variable.

are not necessarily consistent with each other. Concavity of I between input and output, under different sensing schemes, in Poisson channel and Gaussian channel is shown to be concave w.r.t given time when total time is divided equally into any of the four group of sensing time-proportions. No optimal sensing scheme for any of the two channels is investigated in this work.

4.1 Introduction

This chapters considers an experimental design problem of setting, the time-proportions for identifying a four-long binary random vector that is passed through a vector Poisson and vector Gaussian channels, and then based on the observation vector; classification of the input vector is performed and performance of any sensing scheme is then compared. Since, finding the optimal solution for the problem requires computations to be performed in a 15 dimensional search-space and closed-form solutions don not exist, we have instead restricted our search to a reduced dimensional search-space and studied some sensing techniques that are sub-optimal.

The first problem is: Does there exist a configuration among these four configurations which is always performing the best for any given time T and prior p? To answer this we first fixed p, and then we consider T as a free parameter and compute both the mutual information and Bayes probability of total correct detections, for a given set

of parameters, and searched if there exist any instance for which one configuration is the best for some time and then another configuration becomes the best and so on. From mutual information perspective: it is computationally observed that when prior $p < 0.5$ then depending on the value of T any of the four schemes can be better over the others however when $p \geq 0.5$ it is the individual sensing that works best. However, from the detection perspective this is not the case as indicated in fig. (4.3). It is further shown that in each configuration mutual information is concave in T.

The second problem: Does there exist a *hybrid* sensing mechanism that performs better than any of the above four configurations for fixed time T? A hybrid sensing is one when given time T is divide into any one of the four sensing configurations and joint sensing according to the proportion: $(1-\alpha)\cdot T$ and $\alpha \cdot T$ where $0 \leq \alpha \leq 1$, respectively. It turned out that if prior $p \geq 0.5$ then irrespective of other model parameters; individual sensing is the best among any other configurations. For p close to zero hybrid sensing is better over any other as indicated in fig. (4.4). A concavity of mutual information w.r.t α is observed, but no proof is given.

For the vector Gaussian model we have a fixed unit covariance matrix and input X only affects the mean vector. Replace all T_i with $\sqrt{T_i}$ in Poisson model; we have the mean vector for Gaussian channel. It is found that triplet-sensing almost always outperforms any other configuration, irrespective of model parameters. This is shown in fig. (4.5) and fig. (4.6). It is shown that mutual information is concave in

T for any of the four configurations; further in hybrid sensing the mutual information remains concave in α. However, Bayes probability of total correct detection is not necessarily consistent with mutual information results.

4.2 Vector Poisson and Gaussian Channels

4.2.1 Vector Poisson Channel

We consider the vector Poisson channel model [11]:

$$\begin{aligned} \mathrm{Pois}(Y;\Phi X) = P_{Y|X}(Y|X) &= \prod_{i=1}^{15} P_{Y_i|X}(Y_i|X) \\ &= \prod_{i=1}^{15} \mathrm{Pois}(Y_i;(\Phi X)_i) \end{aligned} \tag{4.1}$$

where $\mathrm{Pois}(U;z)$ denotes the standard Poisson distribution of random variable U with parameter z.

We assume input $X = (X_1, X_2, X_3, X_4) \in \{\lambda_0, \lambda_1\}^4$ such that $0 \leq \lambda_0 < \lambda_1$, each X_i is independent and identically distributed with a pmf: $p_{X_i}(x_i = \lambda_0) = 1 - p$ and

$p_{X_i}(x_i = \lambda_1) = p$. $Y = (Y_1, Y_2, \cdots Y_{15}) \in \mathbb{Z}_+^{15}$ and

$$\Phi = \begin{bmatrix} T_1 & 0 & 0 & 0 \\ 0 & T_2 & 0 & 0 \\ 0 & 0 & T_3 & 0 \\ 0 & 0 & 0 & T_4 \\ T_5 & T_5 & 0 & 0 \\ T_6 & 0 & T_6 & 0 \\ T_7 & 0 & 0 & T_7 \\ 0 & T_8 & T_8 & 0 \\ 0 & T_9 & 0 & T_9 \\ 0 & 0 & T_{10} & T_{10} \\ T_{11} & T_{11} & T_{11} & 0 \\ T_{12} & T_{12} & 0 & T_{12} \\ T_{13} & 0 & T_{13} & T_{13} \\ 0 & T_{14} & T_{14} & T_{14} \\ T_{15} & T_{15} & T_{15} & T_{15} \end{bmatrix}. \tag{4.2}$$

The conditional distribution of vector Y given X is a multivariate Poisson distribution:

$$Y \Big| \Big(X = (x_1\ x_2\ x_3\ x_4) \Big) \sim \text{Pois} \left(\begin{bmatrix} T_1 \cdot x_1 \\ T_2 \cdot x_2 \\ T_3 \cdot x_3 \\ T_4 \cdot x_4 \\ T_5 \cdot (x_1 + x_2) \\ T_6 \cdot (x_1 + x_3) \\ T_7 \cdot (x_1 + x_4) \\ T_8 \cdot (x_2 + x_3) \\ T_9 \cdot (x_2 + x_4) \\ T_{10} \cdot (x_3 + x_4) \\ T_{11} \cdot (x_1 + x_2 + x_3) \\ T_{12} \cdot (x_1 + x_2 + x_4) \\ T_{13} \cdot (x_1 + x_3 + x_4) \\ T_{14} \cdot (x_2 + x_3 + x_4) \\ T_{15} \cdot (x_1 + x_2 + x_3 + x_4) \end{bmatrix} \right), \tag{4.3}$$

We define mutual information $I(X;Y)$ as

$$I(X;Y) = H(Y) - H(Y|X), \tag{4.4}$$

where $H(Y)$ is an entropy of a finite Poisson mixture model given as

$$H(Y) = -\sum_{y_1=-\infty}^{\infty}\sum_{y_2=-\infty}^{\infty}\cdots\cdots\sum_{y_{15}=-\infty}^{\infty} P(Y)\cdot \mathrm{Log}_2[P(Y)] \tag{4.5}$$

where

$$\begin{aligned}
&P(Y) = \\
&\sum_{i=1}^{16}\Bigg(P_{X_i}(X_i = x_1, x_2, x_3, x_4)\cdot \mathrm{Pois}\Big(y_1; T_1x_1\Big)\cdot \\
&\mathrm{Pois}\Big(y_2; T_2x_2\Big)\cdot \mathrm{Pois}\Big(y_3; T_3x_3\Big)\cdot \mathrm{Pois}\Big(y_4; T_4x_4\Big)\cdot \mathrm{Pois}\Big(y_5; T_5(x_1+x_2)\Big)\cdot \\
&\mathrm{Pois}\Big(y_6; T_6(x_1+x_3)\Big)\cdot \mathrm{Pois}\Big(y_7; T_7(x_1+x_4)\Big)\cdot \mathrm{Pois}\Big(y_8; T_8(x_2+x_3)\Big)\cdot \\
&\mathrm{Pois}\Big(y_9; T_9(x_2+x_4)\Big)\cdot \mathrm{Pois}\Big(y_{10}; T_{10}(x_3+x_4)\Big)\cdot \mathrm{Pois}\Big(y_{11}; T_{11}(x_1+x_2+x_3)\Big)\cdot \\
&\mathrm{Pois}\Big(y_{12}; T_{12}(x_1+x_2+x_4)\Big)\cdot \mathrm{Pois}\Big(y_{13}; T_{13}(x_1+x_3+x_4)\Big)\cdot \\
&\mathrm{Pois}\Big(y_{14}; T_{14}(x_2+x_3+x_4)\Big)\cdot \mathrm{Pois}\Big(y_{15}; T_{15}(x_1+x_2+x_3+x_4)\Big)\Bigg),
\end{aligned} \tag{4.6}$$

and

$$H(Y|X) = -\sum_{y_1=-\infty}^{\infty}\sum_{y_2=-\infty}^{\infty}\cdots\cdots\sum_{y_{15}=-\infty}^{\infty}\sum_{i=1}^{16} P_{X_i}(X_i)\cdot P(Y|X_i)\cdot \mathrm{Log}_2[P(Y|X_i)] \tag{4.7}$$

where $P(Y|X_i) = \prod_{j=1}^{15}\mathrm{Pois}(Y_j; (\Phi X)_i)$.

Theorem 4.1 *$I(X_1, X_2, X_3, X_4; Y_1, Y_2, Y_3\cdots Y_{15})$ is symmetric in variable-groups:* $(T_1,$

$T_2, T_3, T_4)$*;* $(T_5, T_6, T_7, T_8, T_9, T_{10})$*; and* $(T_{11}, T_{12}, T_{13}, T_{14})$.

Proof: Mutual information $I(X;Y)$ given in (4.4) is invariant under any permutation of variables belonging to the same group. That means interchanging the variables within the same group leaves the expression unchanged. ■

4.2.1.1 Unconstrained objective

For a vector Poisson channel with given prior p, λ_0 and λ_1, which of the following four methods are better over the others when each expression is a function of T solely,

$$I(X_1, X_2, X_3, X_4; \overbrace{Y_1, Y_2, Y_3, Y_4}^{4-\text{Singlets}}) \qquad \text{s.t.} \qquad T_1 = T_2 = T_3 = T_4 = \frac{T}{4} \tag{4.8}$$

$$I(X_1, X_2, X_3, X_4; \overbrace{Y_5, Y_6, Y_7, Y_8, Y_9, Y_{10}}^{6-\text{Pairs}}) \quad \text{s.t.} \quad T_5 = T_6 = T_7 = T_8 = T_9 = T_{10} = \frac{T}{6} \tag{4.9}$$

$$I(X_1, X_2, X_3, X_4; \overbrace{Y_{11}, Y_{12}, Y_{13}, Y_{14}}^{4-\text{Triplets}}) \qquad \text{s.t.} \qquad T_{11} = T_{12} = T_{13} = T_{14} = \frac{T}{4} \tag{4.10}$$

$$I(X_1, X_2, X_3, X_4; \overbrace{Y_{15}}^{1-\text{Quadruplet}}) \qquad \text{s.t.} \qquad T_{15} = T \tag{4.11}$$

Theorem 4.2 $I(X_1, X_2, X_3, X_4; Y_5, Y_6, Y_7, Y_8, Y_9, Y_{10})$ *s.t.* $T_5 = T_6 = T_7 = T_8 = T_9 = T_{10} = \frac{T}{6}$ *is concave in* T.

Proof: $I(X_1, X_2, X_3, X_4; Y_5, Y_6, Y_7, Y_8, Y_9, Y_{10}) = I((X_1 + X_2), (X_1 + X_3), (X_1 + X_4), (X_2 + X_3), (X_2 + X_4), (X_3 + X_4); Y_5, Y_6, Y_7, Y_8, Y_9, Y_{10})$.

Consider the R.H.S of the above equation. From [10, p. 1315], for a random n-tuple vector $X = (X_1, \cdots X_n)$ and for $T \geq 0$, let $Y = (Y_1, \cdots Y_n)$ be jointly distributed with X such that given X, the components of Y are independent with $Y_i|X \sim \text{Poiss}(T \cdot X_i), 1 \leq i \leq n$. Then $\text{mmle}(T) = E\Big[\sum_{i=1}^{n} l(X_i, E[X_i|Y])\Big]$. Since $\frac{\partial}{\partial T} I(X_i; Y) = \text{mmle}(T)$. Each $E\Big[l(X_i, E[X_i|Y])\Big]$ is concave in T. Sum of concave functions result in another Concave function. ■

Corollary 4.1 *Expressions in (4.8), (4.10) and (4.11) are concave in* T *too.*

Note that expressions in (4.8), (4.9) and (4.10), have a tight upper bound of $H(X_1, X_2, X_3, X_4)$ as $T \to \infty$ since the corresponding mappings: from X to $6-$pairs and from X to $4-$triplets are invertible. Whereas, the expression (4.11) has a tight upper bound of $H(\sum_{i=1}^{4} X_i)$ when $T \to \infty$, the mapping from $(X_1, X_2, X_3, X_4) \mapsto \sum_{i=1}^{4} X_i$ is non-invertible.

4.2.1.2 Constrained objective

The second objective is to determine which of the following three configurations are better over the others for a given prior p, λ_0, λ_1 and given fixed time T i.e.,

$$\text{Config}-1: I(X; \overbrace{Y_1, Y_2, Y_3, Y_4, Y_{15}}^{4-\text{Singlets}+1-\text{Quadruplet}}) \quad \text{s.t. } T_1 = T_2 = T_3 = T_4 = \frac{T-\alpha}{4}, T_{15} = \alpha. \tag{4.12}$$

$$\text{Config}-2: I(X; \overbrace{Y_5, Y_6, Y_7, Y_8, Y_9, Y_{10}, Y_{15}}^{6-\text{Pairs}+1-\text{Quadruplet}}) \text{ s.t. } T_5 = T_6 = T_7 = T_8 = T_9 = T_{10} = \frac{T-\alpha}{6},$$

$$T_{15} = \alpha. \tag{4.13}$$

$$\text{Config}-3: I(X; \overbrace{Y_{11}, Y_{12}, Y_{13}, Y_{14}, Y_{15}}^{4-\text{Triplets}+1-\text{Quadruplet}}) \text{ s.t. } T_{11} = T_{12} = T_{13} = T_{14} = \frac{T-\alpha}{4}, T_{15} = \alpha. \tag{4.14}$$

where $0 \leq \alpha \leq T$.

4.2.2 Vector Gaussian Channel

We consider the vector Gaussian channel model as defined in [11] i.e., $Y|X \sim \mathcal{N}(\Phi X, \mathrm{I})$. For a scalar Gaussian channel $Y = \sqrt{T} \cdot X + W$ with $W \sim \mathcal{N}(0,1)$; $I(X;Y)$ is concave in T for arbitrary input signalling [17]. We extend this scalar

model to the vector case. We may also write as

$$\overbrace{\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \\ Y_5 \\ Y_6 \\ Y_7 \\ Y_8 \\ Y_9 \\ Y_{10} \\ Y_{11} \\ Y_{12} \\ Y_{13} \\ Y_{14} \\ Y_{15} \end{bmatrix}}^{Y} = \overbrace{\begin{bmatrix} \sqrt{T_1} & 0 & 0 & 0 \\ 0 & \sqrt{T_2} & 0 & 0 \\ 0 & 0 & \sqrt{T_3} & 0 \\ 0 & 0 & 0 & \sqrt{T_4} \\ \sqrt{T_5} & \sqrt{T_5} & 0 & 0 \\ \sqrt{T_6} & 0 & \sqrt{T_6} & 0 \\ \sqrt{T_7} & 0 & 0 & \sqrt{T_7} \\ 0 & \sqrt{T_8} & \sqrt{T_8} & 0 \\ 0 & \sqrt{T_9} & 0 & \sqrt{T_9} \\ 0 & 0 & \sqrt{T_{10}} & \sqrt{T_{10}} \\ \sqrt{T_{11}} & \sqrt{T_{11}} & \sqrt{T_{11}} & 0 \\ \sqrt{T_{12}} & \sqrt{T_{12}} & 0 & \sqrt{T_{12}} \\ \sqrt{T_{13}} & 0 & \sqrt{T_{13}} & \sqrt{T_{13}} \\ 0 & \sqrt{T_{14}} & \sqrt{T_{14}} & \sqrt{T_{14}} \\ \sqrt{T_{15}} & \sqrt{T_{15}} & \sqrt{T_{15}} & \sqrt{T_{15}} \end{bmatrix}}^{\Phi} \overbrace{\begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{bmatrix}}^{X} + \overbrace{\begin{bmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \\ N_5 \\ N_6 \\ N_7 \\ N_8 \\ N_9 \\ N_{10} \\ N_{11} \\ N_{12} \\ N_{13} \\ N_{14} \\ N_{15} \end{bmatrix}}^{W}, \tag{4.15}$$

where each X_i is independent and identical distributed (i.i.d) discrete random variable with support $\{\lambda_0, \lambda_1\}$ such that p is the probability of occurrence of λ_1 and $(1-p)$ is the occurrence of λ_0 i.e., probability mass function f of scalar input random

variable X_i is

$$f_{X_i}(x_i) = \begin{cases} p & x = \lambda_1 \\ (1-p) & x = \lambda_0 \end{cases} \tag{4.16}$$

Noise vector W is a multivariate Gaussian with zero mean and identity covariance matrix I; and independent of input X. The constraint on the scaling matrix is $\sum_{i=1}^{15} T_i = T$. The conditional distribution of vector Y given X is a multivariate Gaussian:

$$Y\Big|\Big(X=(x_1\,x_2\,x_3\,x_4)\Big)\sim\mathcal{N}\Bigg(\begin{bmatrix}\sqrt{T_1}\cdot x_1\\ \sqrt{T_2}\cdot x_2\\ \sqrt{T_3}\cdot x_3\\ \sqrt{T_4}\cdot x_4\\ \sqrt{T_5}\cdot(x_1+x_2)\\ \sqrt{T_6}\cdot(x_1+x_3)\\ \sqrt{T_7}\cdot(x_1+x_4)\\ \sqrt{T_8}\cdot(x_2+x_3)\\ \sqrt{T_9}\cdot(x_2+x_4)\\ \sqrt{T_{10}}\cdot(x_3+x_4)\\ \sqrt{T_{11}}\cdot(x_1+x_2+x_3)\\ \sqrt{T_{12}}\cdot(x_1+x_2+x_4)\\ \sqrt{T_{13}}\cdot(x_1+x_3+x_4)\\ \sqrt{T_{14}}\cdot(x_2+x_3+x_4)\\ \sqrt{T_{15}}\cdot(x_1+x_2+x_3+x_4)\end{bmatrix},\mathrm{I}\Bigg),\tag{4.17}$$

where I is an identity matrix of size 15×15.

We define mutual information $I(X;Y)$ as

$$I(X;Y)=H(Y)-H(Y|X),\tag{4.18}$$

where $H(Y)$ is a differential entropy of a finite Gaussian mixture model (gmm) given

as

$$H(Y) = -\int_{y_1=-\infty}^{\infty}\int_{y_2=-\infty}^{\infty}\cdots\int_{y_{15}=-\infty}^{\infty} P(Y)\cdot \mathrm{Log}_2[P(Y)]\ dy_1\, dy_2\, dy_3 \cdots dy_{15}, \tag{4.19}$$

where

$$\begin{aligned} P(Y) = \sum_{i=1}^{16}\Bigg(& P_{X_i}(X_i = x_1, x_2, x_3, x_4)\cdot \mathcal{N}\Big(y_1; \sqrt{T_1}x_1, 1\Big)\cdot \mathcal{N}\Big(y_2; \sqrt{T_2}x_2, 1\Big)\cdot \\ & \mathcal{N}\Big(y_3; \sqrt{T_3}x_3, 1\Big)\cdot \mathcal{N}\Big(y_4; \sqrt{T_4}x_4, 1\Big)\cdot \mathcal{N}\Big(y_5; \sqrt{T_5}(x_1+x_2), 1\Big)\cdot \\ & \mathcal{N}\Big(y_6; \sqrt{T_6}(x_1+x_3), 1\Big)\cdot \mathcal{N}\Big(y_7; \sqrt{T_7}(x_1+x_4), 1\Big)\cdot \mathcal{N}\Big(y_8; \sqrt{T_8}(x_2+x_3), 1\Big)\cdot \\ & \mathcal{N}\Big(y_9; \sqrt{T_9}(x_2+x_4), 1\Big)\cdot \mathcal{N}\Big(y_{10}; \sqrt{T_{10}}(x_3+x_4), 1\Big)\cdot \\ & \mathcal{N}\Big(y_{11}; \sqrt{T_{11}}(x_1+x_2+x_3), 1\Big)\cdot \mathcal{N}\Big(y_{12}; \sqrt{T_{12}}(x_1+x_2+x_4), 1\Big)\cdot \\ & \mathcal{N}\Big(y_{13}; \sqrt{T_{13}}(x_1+x_3+x_4), 1\Big)\cdot \mathcal{N}\Big(y_{14}; \sqrt{T_{14}}(x_2+x_3+x_4), 1\Big)\cdot \\ & \mathcal{N}\Big(y_{15}; \sqrt{T_{15}}(x_1+x_2+x_3+x_4), 1\Big)\Bigg), \end{aligned} \tag{4.20}$$

and

$$H(Y|X) = 15 \times 0.5 \times \mathrm{Log}_2[2\pi\, e]. \tag{4.21}$$

The multidimensional integral defined in (4.19) have no closed-form solution, and therefore we need to resort to the Monte Carlo method. The following method is used to numerically evaluate the integral using sampling from a finite Gaussian

mixture.

$$\begin{aligned} H(Y) &= E[-\operatorname{Log}_2[P_Y(Y)]] \\ &\approx -\frac{\sum_i \operatorname{Log}_2[P_Y(s_i)]}{N_s}, \end{aligned} \tag{4.22}$$

Where $P_Y(\cdot)$ is the mixture probability distribution of Y, N_s is the number of MC samples and s_i is the i^{th} sample from multivariate Gaussian mixture distribution.

Theorem 4.3 *$I(X_1, X_2, X_3, X_4; Y_1, Y_2, Y_3 \cdots Y_{15})$ is symmetric in variable-groups: (T_1, T_2, T_3, T_4); $(T_5, T_6, T_7, T_8, T_9, T_{10})$; and $(T_{11}, T_{12}, T_{13}, T_{14})$.*

Proof: Mutual information $I(X;Y)$ given in (4.18) is invariant under any permutation of variables belonging to the same group. That means interchanging the variables within the same group leaves the expression unchanged. ■

4.2.2.1 Unconstrained objective

For a vector Gaussian channel with given prior p, λ_0 and λ_1, which of the following four methods are better over the others when each expression is a function of T

solely,

$$I(X_1, X_2, X_3, X_4; \overbrace{Y_1, Y_2, Y_3, Y_4}^{4-\text{Singlets}}) \qquad \text{s.t.} \qquad T_1 = T_2 = T_3 = T_4 = \frac{T}{4} \tag{4.23}$$

$$I(X_1, X_2, X_3, X_4; \overbrace{Y_5, Y_6, Y_7, Y_8, Y_9, Y_{10}}^{6-\text{Pairs}}) \quad \text{s.t.} \quad T_5 = T_6 = T_7 = T_8 = T_9 = T_{10} = \frac{T}{6} \tag{4.24}$$

$$I(X_1, X_2, X_3, X_4; \overbrace{Y_{11}, Y_{12}, Y_{13}, Y_{14}}^{4-\text{Triplets}}) \qquad \text{s.t.} \qquad T_{11} = T_{12} = T_{13} = T_{14} = \frac{T}{4} \tag{4.25}$$

$$I(X_1, X_2, X_3, X_4; \overbrace{Y_{15}}^{1-\text{Quadruplet}}) \qquad \text{s.t.} \qquad T_{15} = T \tag{4.26}$$

Theorem 4.4 *$I(X_1, X_2, X_3, X_4; Y_5, Y_6, Y_7, Y_8, Y_9, Y_{10})$ in (4.24) is concave in T.*

Proof: It is noted in [7, Theorem 5] that mutual information is a concave function of the squared singular values ($\boldsymbol{\lambda}$) of the precoder matrix P if the first $m^{'}$ eigenvectors of the channel covariance matrix ($\mathbf{R}_H = H^{\intercal}\mathbf{R}_Z^{-1}H$) coincide with the left singular vectors of the precoder P i-e $\mathsf{H}_{\lambda} I(S;Y) \leq 0$ for the signal model $Y = HPS + Z$ where $H \in \mathbb{R}^{n \times p}$ is the channel, S is the input signaling $S \in \mathbb{R}^m$, P is a precoder matrix $P \in \mathbb{R}^{p \times m}$ and $Z \in \mathbb{R}^n$ is Gaussian noise independent of the input S and has

covariance matrix $\mathbf{R}_Z$.

For our problem: $H = I$, $\mathbf{R}_Z^{-1} = \Lambda = I$, $P = \Phi$, $S = X$ and $Z = W$. The singular value decomposition of $\Phi = U\Sigma V^*$.

By substituting $T_1 = T_2 = T_3 = T_4 = 0$, $T_5 = T_6 = \cdots T_{10} = T$ and $T_{15} = 0$ in (4.15), the squared singular values of Φ are $[\lambda_1, \lambda_2 \cdots \lambda_{15}] = [6T, 2T, 2T, 2T, 0, 0, 0, \cdots 0]$. This is just the composition with an affine transformation on the domain. ■

Corollary 4.2 *$I(X; Y_1, Y_2, Y_3, Y_4)$, $I(X; Y_{11}, Y_{12}, Y_{13}, Y_{14})$ and $I(X; Y_{15})$ are concave in T, since squared singular values are $[T, T, T, T, 0, 0 \cdots 0]$, $[9T, 2T, 2T, 2T, 0, 0, 0, \cdots 0]$ and $[4T, 0, 0, 0, \cdots 0]$ respectively.*

4.2.2.2 Constrained objective

The second objective for the vector Gaussian channel is which of the following three configurations are better over the others for a given prior p, λ_0, λ_1 and given fixed

time T i.e.,

$$\text{Config}-1: I(X; \overbrace{Y_1, Y_2, Y_3, Y_4, Y_{15}}^{4-\text{Singlets}+1-\text{Quadruplet}}) \quad \text{s.t. } T_1 = T_2 = T_3 = T_4 = \frac{T-\alpha}{4}, T_{15} = \alpha. \tag{4.27}$$

$$\text{Config}-2: I(X; \overbrace{Y_5, Y_6, Y_7, Y_8, Y_9, Y_{10}, Y_{15}}^{6-\text{Pairs}+1-\text{Quadruplet}}) \text{ s.t. } T_5 = T_6 = T_7 = T_8 = T_9 = T_{10} = \frac{T-\alpha}{6}, \quad T_{15} = \alpha. \tag{4.28}$$

$$\text{Config}-3: I(X; \overbrace{Y_{11}, Y_{12}, Y_{13}, Y_{14}, Y_{15}}^{4-\text{Triplets}+1-\text{Quadruplet}}) \text{ s.t. } T_{11} = T_{12} = T_{13} = T_{14} = \frac{T-\alpha}{4}, T_{15} = \alpha. \tag{4.29}$$

where $0 \leq \alpha \leq T$.

Theorem 4.5 *$I(X_1, X_2, X_3, X_4; Y_5, Y_6, Y_7, Y_8, Y_9, Y_{10}, Y_{15})$ s.t. $T_5 = T_6 = T_7 = T_8 = T_9 = T_{10} = \frac{T-\alpha}{6}, T_{15} = \alpha$, where $0 \leq \alpha \leq T$, is concave in α.*

Proof: We again resort to the [7, Theorem 5].

For our problem: $H = I$, $\mathbf{R}_Z^{-1} = \Lambda = I$, $P = \Phi$, $S = X$ and $Z = W$. The singular value decomposition of $\Phi = U\Sigma V^*$.

By substituting $T_1 = T_2 = T_3 = T_4 = 0$, $T_5 = T_6 = \cdots T_{10} = \frac{T-\alpha}{2}$ and $T_{15} = \alpha$ in (4.15), the squared singular values of Φ are $[\lambda_1, \lambda_2 \cdots \lambda_{15}] =$

$[0, 0 \cdots 0, \frac{T-\alpha}{3}, \frac{T-\alpha}{3}, \frac{T-\alpha}{3}, T + 3\alpha]$. This is just the composition with an affine transformation on the domain. Concavity remains preserved under affine transformation [12, page 79-86]. ■

Corollary 4.3 *$I(X; Y_1, Y_2, Y_3, Y_4, Y_{15})$ s.t. $T_1 = T_2 = T_3 = T_4 = \frac{T-\alpha}{4}, T_{15} = \alpha$ and $I(X; Y_{11}, Y_{12}, Y_{13}, Y_{14}, Y_{15})$ s.t. $T_{11} = T_{12} = T_{13} = T_{14} = \frac{T-\alpha}{4}, T_{15} = \alpha$ are concave in α, since squared singular values are $[\lambda_1, \lambda_2 \cdots \lambda_{15}] = [0, 0 \cdots 0, \frac{T-\alpha}{4}, \frac{T-\alpha}{4}, \frac{T-\alpha}{4}, \frac{T+15\alpha}{4}]$ and $[\lambda_1, \lambda_2 \cdots \lambda_{15}] = [0, 0 \cdots 0, \frac{T-\alpha}{4}, \frac{T-\alpha}{4}, \frac{T-\alpha}{4}, \frac{9T+7\alpha}{4}]$, respectively.*

Therefore, the constraint objectives in (4.27), (4.28) and (4.29) are all concave in α.

4.3 Detection Theoretic Description

4.3.1 Bayes criterion

In terms of Bayes detection we may consider the problem as deciding among the $16-$ hypotheses $(\mathcal{H}_i, 1 \leq i \leq 16)$ for a fixed time-proportions. Considering the prior probability of each hypothesis as π_i such that $\sum_{i=1}^{16} \pi_i = 1$. Let C_{il} is the cost of deciding $\mathcal{D}_i$ when $\mathcal{H}_l$ is correct, then the average cost is $r = \sum_{i=1}^{16} \sum_{l=1}^{16} \pi_l C_{il} P_{il}$, where P_{il} is the probability of deciding $\mathcal{D}_i$ when $\mathcal{H}_l$ is true.

For Gaussian problem with fixed sensing-time proportions; $\mathcal{H}_i : \mathcal{N}_{15}(\mu_i, \mathbb{I})$, with prior $\pi_i, \quad 1 \leq i \leq 16$. Where $\mathcal{N}_{15}(\mu_i, \mathbb{I})$ is a $15-$dimensional multivariate normal distribution with fixed covariance unit-matrix and $15-$component random mean vector μ_i. $P_{il} = \int_{y \in \mathcal{R}_i} P_l(y_1, y_2 \cdots y_{15} | \mathcal{H}_i) \partial y$. We only consider the MAP criterion where cost is

$$C_{il} = \begin{cases} 0 & i = l \\ 1 & i \neq l \end{cases} \tag{4.30}$$

This simplifies the detection rule to deciding:

$$\mathcal{D}_i : \quad \text{if} \quad \pi_i p_i(y_1, y_2, \cdots y_{15} | \mathcal{H}_i) \geq \pi_n p_n(y_1, y_2, \cdots y_{15} | \mathcal{H}_n) \text{ for all } n \neq i \tag{4.31}$$

and for any fixed time-proportions $(T_1, T_2, \cdots T_{15})$ under consideration.

For Poisson problem with fixed sensing-times: $\mathcal{H}_i : \text{Poiss}_{15}(\mu_i)$, with prior $\pi_i, \quad 1 \leq i \leq 16$. Where $\text{Pois}_{15}(\mu_i) = \prod_{i=1}^{15} \text{Pois}((\Phi X)_i)$ is a $15-$dimensional multivariate Poisson distribution with $15-$ component random mean vector μ_i. $P_{il} = \sum_{y \in \mathcal{R}_i} P_l(y_1, y_2 \cdots y_{15} | \mathcal{H}_i)$. Thus we are interested in minimizing the Bayes risk r(under both constrained and unconstrained objectives defined above) for any given structure in time-proportions $(T_1, T_2 \cdots T_{15})$ i-e

$$\min_{(T_1, T_2, \cdots T_{15})} r \quad \text{s.t.} \quad \sum_{i=1}^{15} T_i = T. \tag{4.32}$$

Equivalently, we may write

$$\max_{(T_1,T_2,\cdots T_{15})} P_d \quad \text{s.t.} \ \sum_{i=1}^{15} T_i = T \tag{4.33}$$

where P_d is probability of total correct detections, $P_d = 1 - r$.

Conjecture 4.1 *The optimal solution of finding the best time-proportions in a given fixed time T, both under information theoretic and detection theoretic metrics, has a specific structure:* $(T_1 := a, T_2 := a, T_3 := a, T_4 := a, T_5 := b, T_6 := b, T_7 := b, T_8 := b, T_9 := b, T_{10} := b, T_{11} := c, T_{12} := c, T_{13} := c, T_{14} := c, T_{15} := d)$. *Where* $0 \leq a, b, c, d \leq T$ *and* $4 \cdot a + 6 \cdot b + 4 \cdot c + d = T$.

4.4 Computational setup

To compute the mutual information expressions for the Poisson channel given in (4.8)-(4.11) and for the Gaussian channel given in (4.23)-(4.26), we have utilized a Monte Carlo method. For any of the time settings, under any sensing scheme, we first generate 10^6 samples from the respective Poisson mixture pmf (or Gaussian mixture pdf). These samples are generated in a manner that based on the prior of each component Poisson multivariate (or component Gaussian multivariate), we took the same percent of samples from that component. Further, as in each component the random variables are mutually independent, this simplifies the samples' generation from any component. After the samples are generated from any component, for fixed time-proportions $(T_1, T_2, \cdots T_{15})$ and given model parameters $(\lambda_0, \lambda_1, T, p)$, we calculated the $P(Y)$ as given in (4.6) and (4.20). From the computational-time point-of-view, this calculation of $P(Y)$ is most time-consuming than any other step and this is due to the calculation of sixteen $15-$dimensional multivariate components involved in mixture distribution functions. Log_2 is then taken of the 10^6 points of $P(Y)$ before taking the average as given in (4.22). Once we calculated the $H(Y)$, then comes the conditional entropy $H(Y|X)$. For the conditional Gaussian entropy the expression is simple as given in (4.21). For the conditional Poisson entropy we first truncate the conditional Poisson pmfs of each variable Y_i to a sufficiently large value which is calculated as $y_{i_{max}} = 2 \cdot \mathrm{PoissCDF}^{-1}(1 - 1.110223024625157 \cdot 10^{-16}, \lambda)$,

where $y_{i_{max}}$ is the truncation point and $\mathrm{PoissCDF}^{-1}(m, \lambda)$ is the inverse Poisson cumulative distribution function (cdf) with parameter λ and at point m. After the truncation; a finite summation for the individual variable y_i can be calculated easily i-e $H(Y_i|\lambda) = -\sum_{j=0}^{y_{i_{max}}} \mathrm{Log}_2[\mathrm{Poiss}(j;\lambda)] \cdot \mathrm{Poiss}(j;\lambda)$. Conditional entropy for Poisson channel can then be readily calculated from (4.7). For the MAP detection, we use the same samples, for posterior probabilities of each of the $15-$hypothesis, that are previously used for the calculation of mutual information. We had generated the samples from each of the component with the same proportion as defined by the prior of that component and then calculated the joint probability of that sample point Y_s with each hypothesis. Deciding in favor of the hypothesis for which the maximum of the joint probability $P(Y_s, \mathcal{H}_i)$ happens among 16 such probabilities; as given in (4.31). The computed results for mutual information and Bayes probability of total correct detections are shown in fig. (4.3), (4.4), (4.5), and (4.6).

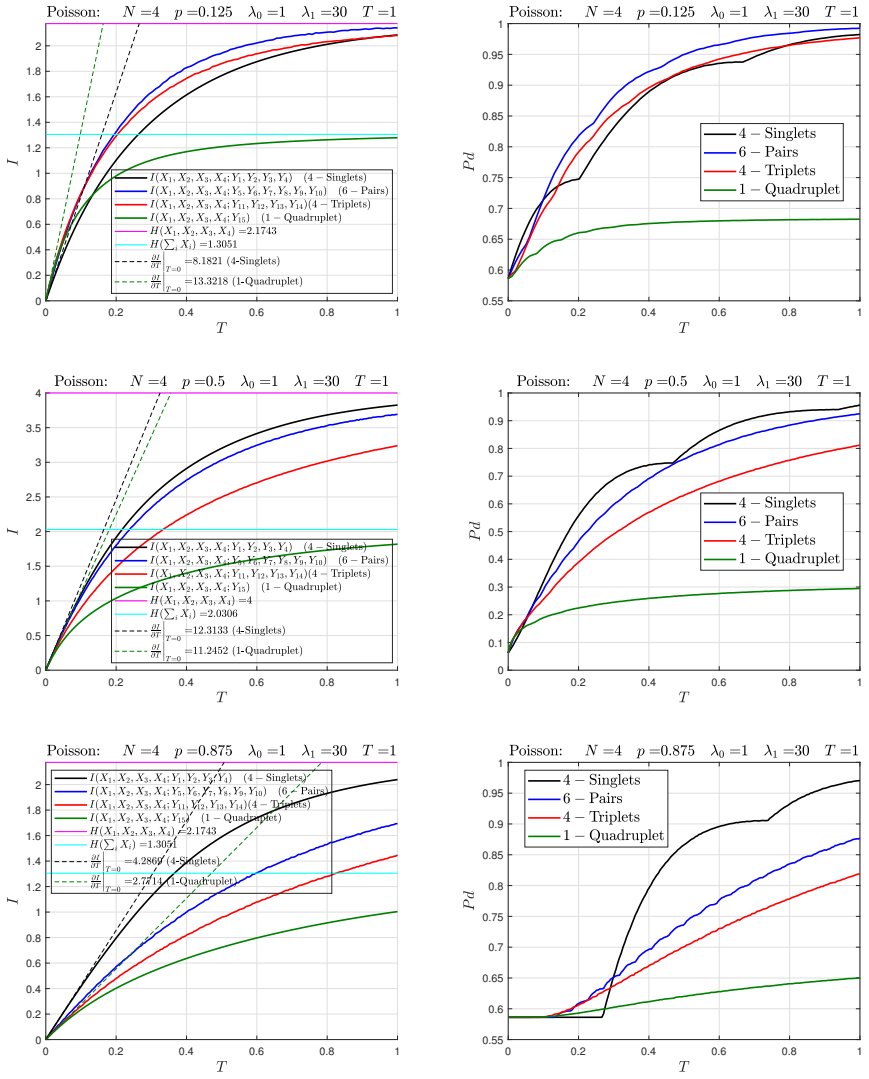

Figure 4.3: Poisson channel: (Left) $I(X;Y)$ vs. T, (right) P_d vs. T for varying *prior* probabilities p.
(MATLAB-CODE F.13 & F.18.7)

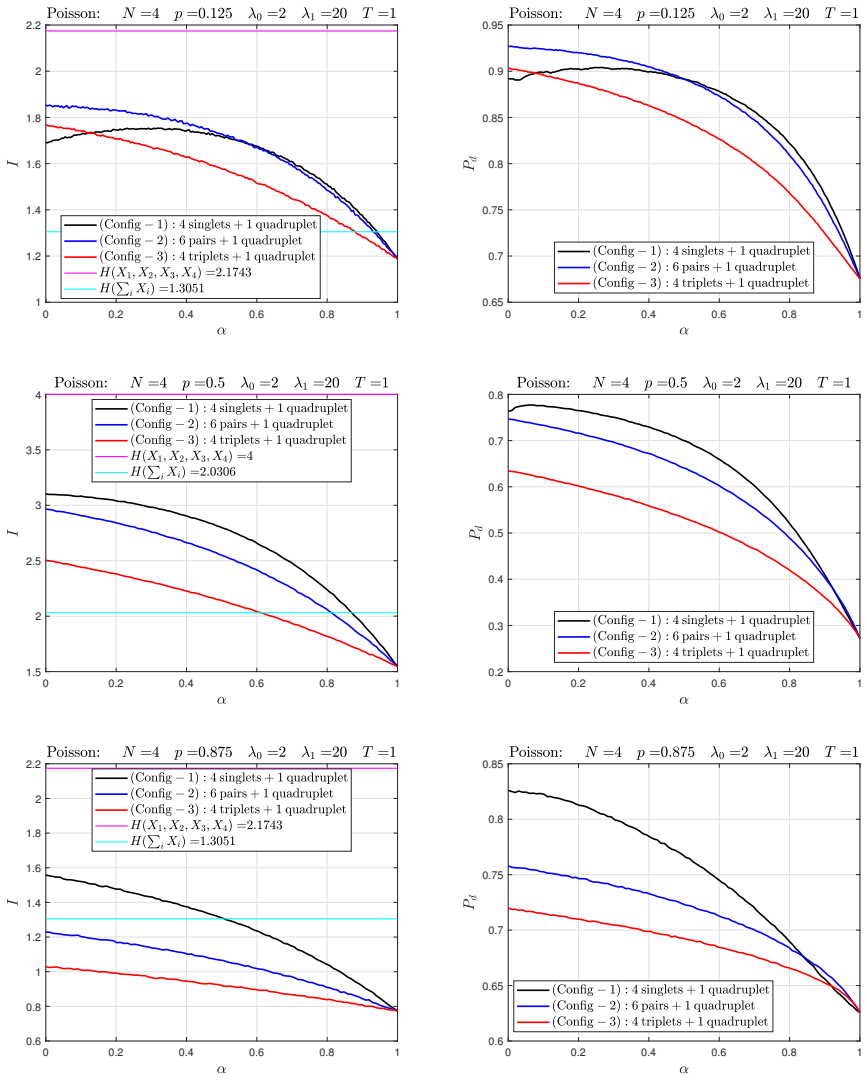

Figure 4.4: Poisson channel :
$\text{Config}-1:\left(\frac{\text{T}-\alpha}{4},\frac{\text{T}-\alpha}{4},\frac{\text{T}-\alpha}{4},\frac{\text{T}-\alpha}{4},0,0,0,0,0,0,0,0,0,0,0,\alpha\right)$;
$\text{Config}-2:\left(0,0,0,0,\frac{\text{T}-\alpha}{6},\frac{\text{T}-\alpha}{6},\frac{\text{T}-\alpha}{6},\frac{\text{T}-\alpha}{6},\frac{\text{T}-\alpha}{6},\frac{\text{T}-\alpha}{6},0,0,0,0,\alpha\right)$ and
$\text{Config}-3:\left(0,0,0,0,0,0,0,0,0,0,0,\frac{\text{T}-\alpha}{4},\frac{\text{T}-\alpha}{4},\frac{\text{T}-\alpha}{4},\frac{\text{T}-\alpha}{4},\alpha\right)$ where
$0 \leq \alpha \leq T$ and time constraint $\sum_{i=1}^{15} T_i = T$ for $\lambda_0 = 2$, $\lambda_1 = 20$, and varying *prior* probability p.
(MATLAB-CODE F.14 & F.18.7)

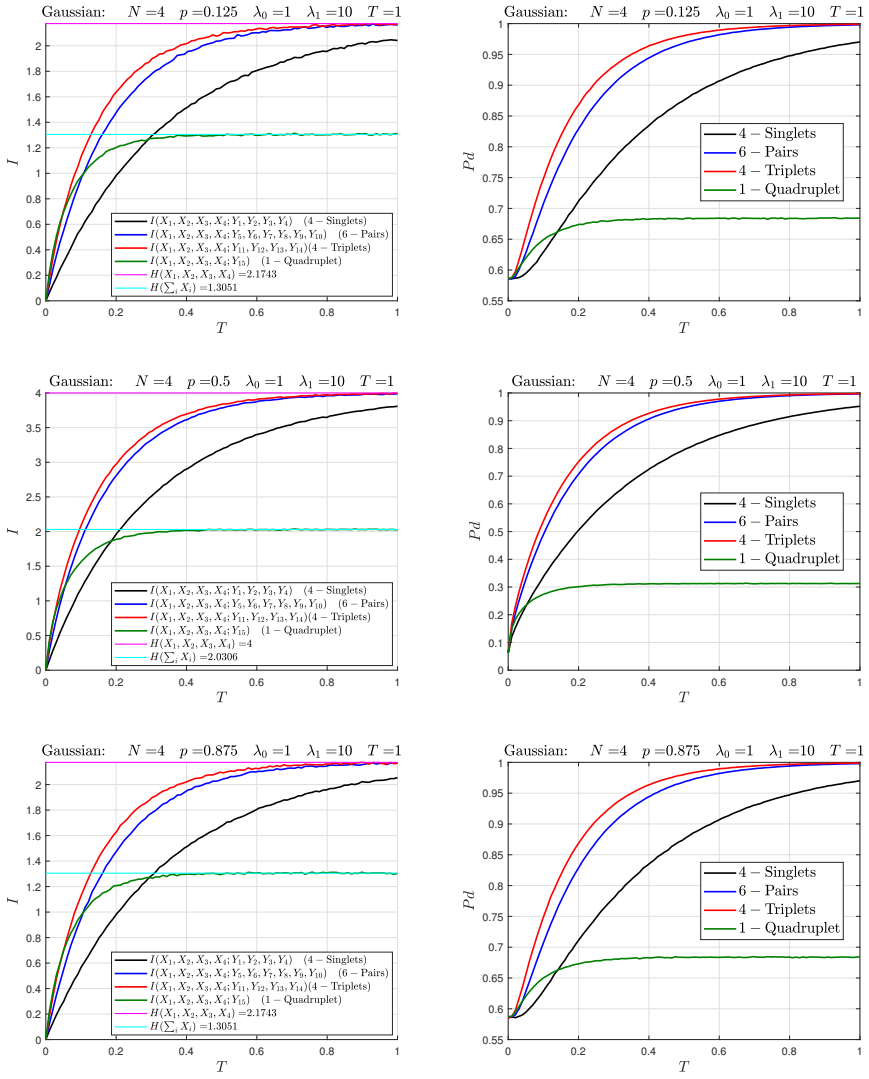

Figure 4.5: Gaussian channel: (Left) $I(X;Y)$ vs. T, (right) P_d vs. T for varying *prior* probabilities p.
(MATLAB-CODE F.15 & F.18.8)

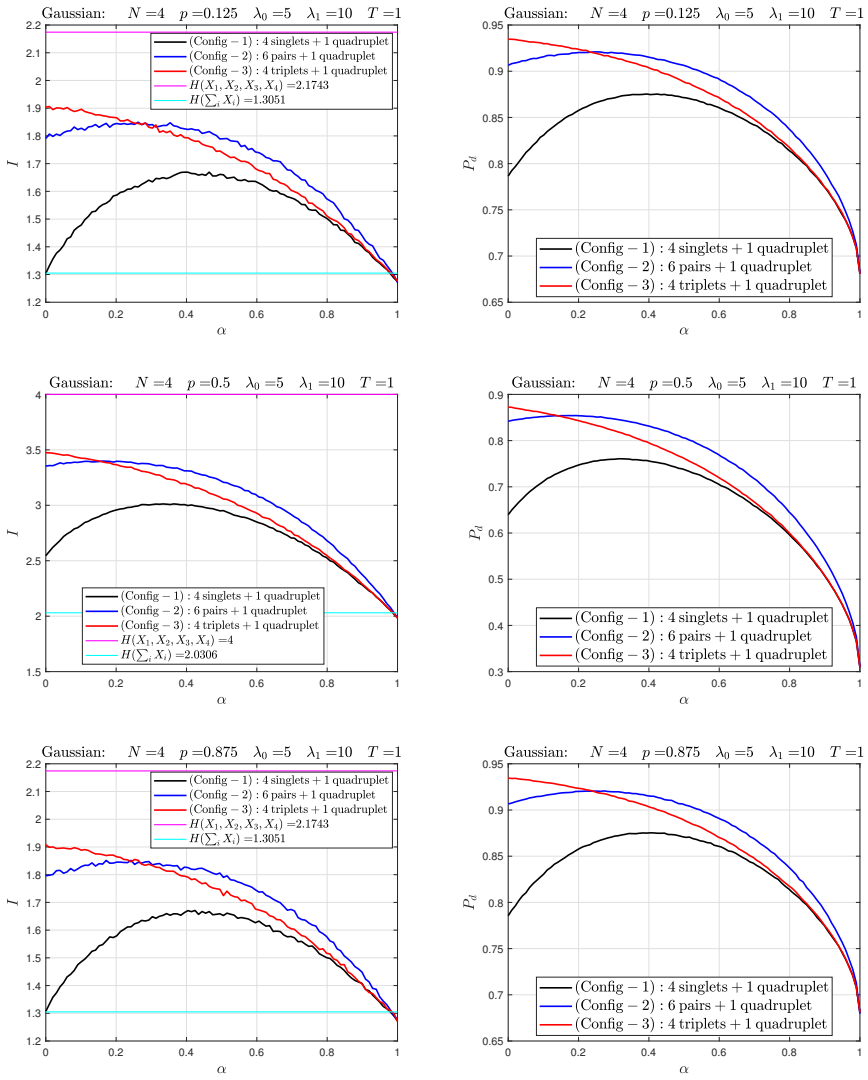

Figure 4.6: Gaussian channel :
$\text{Config}-1:\left(\frac{\mathrm{T}-\alpha}{4},\frac{\mathrm{T}-\alpha}{4},\frac{\mathrm{T}-\alpha}{4},\frac{\mathrm{T}-\alpha}{4},0,0,0,0,0,0,0,0,0,0,0,\alpha\right)$;
$\text{Config}-2:\left(0,0,0,0,\frac{\mathrm{T}-\alpha}{6},\frac{\mathrm{T}-\alpha}{6},\frac{\mathrm{T}-\alpha}{6},\frac{\mathrm{T}-\alpha}{6},\frac{\mathrm{T}-\alpha}{6},\frac{\mathrm{T}-\alpha}{6},0,0,0,0,0,\alpha\right)$ and
$\text{Config}-3:\left(0,0,0,0,0,0,0,0,0,0,0,\frac{\mathrm{T}-\alpha}{4},\frac{\mathrm{T}-\alpha}{4},\frac{\mathrm{T}-\alpha}{4},\frac{\mathrm{T}-\alpha}{4},\alpha\right)$ where
$0 \leq \alpha \leq T$ and time constraint $\sum_{i=1}^{15} T_i = T$ for $\lambda_0 = 5$, $\lambda_1 = 10$, and varying *prior* probability p.
(MATLAB-CODE F.16 & F.18.8)

4.5 Conclusion

In this work, a sensor scheduling problem for four target detection in a vector Poisson and Gaussian channel was considered using metrics of mutual information I and Bayes risk with $0-1$ cost.

First, four sensing schemes: $4-$singlets (individual) sensing; $6-$pairs sensing; $4-$triplets sensing; and $1-$quadruplet sensing were considered with the total given time T being variable. It was shown that mutual information between input and output is concave w.r.t given time, (and irrespective of any other model parameters) for either of the two channels. It is further noted that for the Poisson channel; individual sensing is the best among the four strategies if prior $p \geq 0.5$ from I perspective. However, in the equivalent Bayesian risk minimization problem, neither the concavity of Bayesian probability, $P_{d,}$ of total correct detections w.r.t time is observed nor individual sensing is always found to be the best among others. Whereas for the Gaussian channel it is the $4-$triplets sensing scheme that almost outperform the other three sensing-schemes and Bayesian P_d is not consistent with the I computational results.

Secondly, in another constrained configuration: where total time T is always held fixed while linearly distributed between joint sensing: and individual sensing; $6-$pair

sensing and $4-$ triplets sensing. From computations; concavity of I is observed w.r.t time shifting parameter α. For the Poisson problem from the I perspective; it is again the $4-$individual sensing that outperforms any other configuration for $p \geq 0.5$. This is not very much consistent from the Bayes detection perspective, however. $6-$ pair sensing is more beneficial than $4-$ individual sensing for prior close to zero. In the Gaussian channel it is the $4-$ triplets-sensing scheme that is the best among any other scheme and this is evident from both I and P_d metrics. It is shown that I is concave in α.

The theorems of concavity of I in both constrained and unconstrained objectives for both channels, confirms the observations that are made about the concavity of I in time; that is evident in various computational plots. It would be interesting in figuring out why time-divisions: $T_1 = T_2 = T_3 = T_4$, and $T_5 = T_6 = T_7 = T_8 = T_9 = T_{10}$ and $T_{11} = T_{12} = T_{13} = T_{14}$ are better than being not equal in respective groups?

Chapter 5

Conclusion

This work attempted to address the problem of sensor scheduling in a vector Poisson and Gaussian channels for two targets and four targets detection using criteria of mutual information and Bayesian risk.

For a Poisson channel: based on the computations, it is observed that mutual information and Bayes probability of total correct detections are not monotonic, in general. Mutual information is observed to be concave in counting times proportions, but no proof is given. From a mutual information perspective and for a two target and four target detection problem, it is observed that if the higher intensity of Poisson point process is more likely than lower intensity $p \geq 0.5$ then the optimal sensing method is to count the Poisson arrivals from the two targets individually

irrespective of other model parameters. In contrast, if the lower intensity is more likely $(1-p) \geq 0.5$ than it is the hybrid sensing that is optimal. In the limiting case when $p \to 0$, it is the joint sensing that becomes optimal.

For a Gaussian channel: based on computations for a two target detection case, it is observed that objective functions for maximization of I and P_d does not necessarily lead to the same optimal solution. It is the *hybrid sensing* that is observed to be optimal for any set of model parameters and just based on prior (as in the Poisson channel) it can not be guessed which sensing method would be optimal. Mutual information is observed to be concave in the respective time domain. Only for some special cases of domain of I we have provided the proof of concavity. For a four target detection problem it is found that it is the $4-$ triplets sensing (that is done by forming all possible four triplets of four long vector X) that outperforms any other sensing schemes under investigation. However, search for an optimal sensing scheme for four target detection is not done in this dissertation for the four target problem.

5.0.1 Future work

1. We saw in chapters 2, 3 and 4 that we only assumed the binary input signalling through a vector Poisson and Gaussian channels; and we saw that metrics of

mutual information and Bayesian probability of total correct detections may come out with different solutions when sensing time is short. However, if the sensing time is long enough then both metrics become more and more consistent. This effect might be due to the countably finite alphabet set of input vector X for which detection is sought. It would be interesting to study the same two channels and check the consistency of the two metrics, when the input vector X would have an uncountable finite (or infinite) support. One instance might be to assume a two parameter Beta distribution on each component of X for a uncountable finite support, other instance might be to assume a single parameter exponential distribution on the components of X for an uncountable infinite support, to check the consistency between the two metrics.

2. When dimension of the input vector X approaches to infinity, and the input signalling X_i remains binary then for a given small sensing time in the Poisson channel; it is better to perform the joint sensing instead of the individual sensing [19]. It is further noted that when $N \to \infty$ resulting in the distribution of $\sum_i X_i$ approaching to Poisson distribution then the rate parameter of this input Poisson distribution that would maximize the mutual information is $\lambda \approx 1.35$ for some prior $p* < 0.5$. While this is true for all binary input X_is that are i.i.d, it would be interesting to know when all or any of the X_i have different supports among each other. This question is addressed for a two user

inputs in [20], where this situation is called a *non-symmetric* Poisson multiple access channels.

3. Schur concavity along with time-symmetry, in the constrained problem of maximizing I in time arguments, is sufficient to reduce the exponentially rising dimension of search space (w.r.t dimension of X) to linearly rising search space. Schur concavity is mathematically less tedious than working with concavity for our problem. Gradient of mutual information for both vector Poisson and Gaussian channels are provided in the literature [11], it is suspected that some more useful results can be extracted from this work in the context of our problem. What can be deduced about the possible Quasi-concavity in the constrained problem would be a good direction to explore?

4. For $N > 4$, using the Monte Carlo method of generating the samples from the multivariate Poisson and Gaussian distribution and then estimating the joint entropy of the observations (or computing the posterior probabilities) would become quite slow or ineffective eventually for large N. For $N = 100$ with input X assuming log-normal mixture, in [11] the classification accuracy of designed Φ is compared with that of a randomly design Φ; using the gradient result for the two vector channels. It is done by using the 500 Monte Carlo samples to calculate the gradient first and then 500 iterations for gradient descent. Their algorithm converged well after few iterations. Exploring this method further might help to device some algorithm to search for optimal

solution for our problem in higher dimensional space.

5. It is seen that only one sensor is required for the detection with the joint sensing. The advantage is two fold: first the hardware simplicity (as only one sensor is required); second for short interval of time (with suitable prior of course), joint sensing might be better than any other sensing method. However, joint-sensing fails to resolve the location of the targets no matter how long the sensing time is available. In contrast the individual sensing needs sensors that grows linearly with N, however, it can resolve the target's location with increasing accuracy as given time is increased until there is no ambiguity in the location (as time approaches infinity). Secondly it is not drastically inefficient when compared to the joint sensing when joint sensing is performing the best among any other sensing schemes. The other sensing methods doublets, triplets, etc need sensors that grow exponentially fast w.r.t N and from the hardware design complexity perspective this makes the system designing practically infeasible. We may expect that the *individual sensing* provide the best trade-off in higher dimensions between required number of sensors and the amount of acceptable mutual information (or Bayesian probability of total correct detections) between input X and observations Y for arbitrary model parameters but that needs further evidence and exploration.

References

[1] Adell, J. A.; Lekuona, A.; Yu, Y. *IEEE Transactions on Information Theory* **2010**, *56*(5), 2299–2306.

[2] Hero, A. O.; Castan, D. A.; Cochran, D.; Kastella, K. *Foundations and Applications of Sensor Management;* Springer Publishing Company, Incorporated, 2007.

[3] Lee, H.; Teo, K. L.; Lim, A. E. *Automatica* **2001**, *37*(12), 2017–2023.

[4] Manyika, J. M.; Durrant-Whyte, H. F. In *Applications in Optical Science and Engineering*, pages 202–213. International Society for Optics and Photonics, 1992.

[5] Schmaedeke, W. W. In *Optical Engineering and Photonics in Aerospace Sensing*, pages 156–164. International Society for Optics and Photonics, 1993.

[6] Yang, Y.; Blum, R. S. *IEEE Transactions on Aerospace and Electronic Systems* **2007**, *43*(1).

[7] Payaró, M.; Palomar, D. P. *IEEE Transactions on Information Theory* **2009**, *55*(8), 3613–3628.

[8] Verdú, S. *IEEE Transactions on Information Theory* **2010**, *56*(8), 3712–3720.

[9] Guo, D.; Shamai, S.; Verdú, S. *IEEE Transactions on Information Theory* **2008**, *54*(5), 1837–1849.

[10] Atar, R.; Weissman, T. *IEEE Transactions on Information theory* **2012**, *58*(3), 1302–1318.

[11] Wang, L.; Carlson, D. E.; Rodrigues, M. R.; Calderbank, R.; Carin, L. *IEEE Transactions on Information Theory* **2014**, *60*(5), 2611–2629.

[12] Boyd, S.; Boyd, S. P.; Vandenberghe, L. *Convex optimization;* Cambridge university press, 2004.

[13] Ross, S. M.; others. *Stochastic Processes*, Vol. 2; John Wiley & Sons New York.

[14] Yeung, R. W. *Information theory and network coding;* Springer Science & Business Media, 2008.

[15] Cover, T. M.; Thomas, J. A. *Elements of Information Theory;* John Wiley & Sons, 2012.

[16] Schonhoff, T. A.; Giordano, A. A. *Detection and Estimation Theory and its Applications;* Pearson College Division, 2006.

[17] Guo, D.; Shamai, S.; Verdú, S. *Information Theory, IEEE Transactions on* **2005**, *51*(4), 1261–1282.

[18] Palomar, D. P.; Verdú, S. *IEEE Transactions on Information Theory* **2006**, *52*(1), 141–154.

[19] Lapidoth, A.; Shamai, S. *IEEE Transactions on Information Theory* **1998**, *44*(2), 488–501.

[20] Aisha, A.-u.; Liang, Y.; Lai, L.; Shamai, S. In *2016 IEEE International Symposium on Information Theory (ISIT)*, pages 375–379. IEEE, 2016.

[21] Trefethen, L. N.; Bau III, D. *Numerical Linear Algebra*, Vol. 50; Siam, 1997.

Appendix A

Expression for Mutual Information (Two-Target)

$$
\begin{aligned}
&I(X;Y) \\
&= -\sum_{y_1=0}^{\infty}\sum_{y_2=0}^{\infty}\sum_{y_3=0}^{\infty}\Bigg[\Bigg((1-p)^2\cdot \mathrm{Poiss}(Y_1;\lambda_0 T_1)\cdot \\
&\mathrm{Poiss}(Y_2;\lambda_0 T_2)\cdot \mathrm{Poiss}(Y_3;2\lambda_0 T_3) + p(1-p)\cdot \\
&\mathrm{Poiss}(Y_1;\lambda_0 T_1)\cdot \mathrm{Poiss}(Y_2;\lambda_1 T_2)\cdot \mathrm{Poiss}(Y_3;(\lambda_0+\lambda_1)T_3) + \\
&p(1-p)\cdot \mathrm{Poiss}(Y_1;\lambda_1 T_1)\cdot \mathrm{Poiss}(Y_2;\lambda_0 T_2)\cdot \\
&\mathrm{Poiss}(Y_3;(\lambda_1+\lambda_0)T_3) + p^2\cdot \mathrm{Poiss}(Y_1;\lambda_1 T_1)\cdot \\
&\mathrm{Poiss}(Y_2;\lambda_1 T_2)\cdot \mathrm{Poiss}(Y_3;2\lambda_1 T_3)\Bigg)\cdot
\end{aligned}
$$

$$\Bigg(\text{Log}_2[(1-p)^2 \cdot \text{Poiss}(Y_1; \lambda_0 T_1) \cdot \text{Poiss}(Y_2; \lambda_0 T_2) \cdot$$

$$\text{Poiss}(Y_3; 2\lambda_0 T_3) + p(1-p) \cdot \text{Poiss}(Y_1; \lambda_0 T_1) \cdot$$

$$\text{Poiss}(Y_2; \lambda_1 T_2) \cdot \text{Poiss}(Y_3; (\lambda_0 + \lambda_1) T_3) + p(1-p) \cdot$$

$$\text{Poiss}(Y_1; \lambda_1 T_1) \cdot \text{Poiss}(Y_2; \lambda_0 T_2) \cdot \text{Poiss}(Y_3; (\lambda_1 + \lambda_0) T_3) +$$

$$p^2 \cdot \text{Poiss}(Y_1; \lambda_1 T_1) \cdot \text{Poiss}(Y_2; \lambda_1 T_2) \cdot \text{Poiss}(Y_3; 2\lambda_1 T_3)] \Bigg) \Bigg] -$$

$$\Bigg[\Bigg((1-p)^2 \cdot \sum_{y_1=0}^{\infty} \sum_{y_2=0}^{\infty} \sum_{y_3=0}^{\infty} (\text{Poiss}(Y_1; \lambda_0 T_1) \cdot \text{Poiss}(Y_2; \lambda_0 T_2) \cdot$$

$$\text{Poiss}(Y_3; 2\lambda_0 T_3)) \cdot \text{Log}_2[\text{Poiss}(Y_1; \lambda_0 T_1) \cdot \text{Poiss}(Y_2; \lambda_0 T_2) \cdot$$

$$\text{Poiss}(Y_3; 2\lambda_0 T_3)] \Bigg) + \Bigg(p(1-p) \cdot \sum_{y_1=0}^{\infty} \sum_{y_2=0}^{\infty} \sum_{y_3=0}^{\infty}$$

$$(\text{Poiss}(Y_1; \lambda_0 T_1) \cdot \text{Poiss}(Y_2; \lambda_1 T_2) \cdot \text{Poiss}(Y_3; (\lambda_0 + \lambda_1) T_3) \cdot$$

$$\text{Log}_2[\text{Poiss}(Y_1; \lambda_0 T_1) \cdot \text{Poiss}(Y_2; \lambda_1 T_2) \cdot$$

$$\text{Poiss}(Y_3; (\lambda_0 + \lambda_1) T_3)] \Bigg) + \Bigg(p(1-p) \cdot \sum_{y_1=0}^{\infty} \sum_{y_2=0}^{\infty} \sum_{y_3=0}^{\infty}$$

$$(\text{Poiss}(Y_1; \lambda_1 T_1) \cdot \text{Poiss}(Y_2; \lambda_0 T_2) \cdot \text{Poiss}(Y_3; (\lambda_1 + \lambda_0) T_3) \cdot$$

$$\text{Log}_2[\text{Poiss}(Y_1; \lambda_1 T_1) \cdot \text{Poiss}(Y_2; \lambda_0 T_2) \cdot$$

$$\text{Poiss}(Y_3; (\lambda_1 + \lambda_0) T_3)] \Bigg) + \Bigg(p^2 \cdot \sum_{y_1=0}^{\infty} \sum_{y_2=0}^{\infty} \sum_{y_3=0}^{\infty}$$

$$(\text{Poiss}(Y_1; \lambda_1 T_1) \cdot \text{Poiss}(Y_2; \lambda_1 T_2) \cdot \text{Poiss}(Y_3; 2\lambda_1 T_3) \cdot$$

$$\text{Log}_2[(\text{Poiss}(Y_1; \lambda_1 T_1) \cdot \text{Poiss}(Y_2; \lambda_1 T_2) \cdot \text{Poiss}(Y_3; 2\lambda_1 T_3)] \Bigg) \Bigg].$$

Appendix B

Expression for Bayes Probability of Detection (Two-Target)

$$
\begin{aligned}
&P_d \\
&= \sum_{(y_1,y_2,y_3)} f_{00} : \{(f_{00} > f_{01}) \wedge (f_{00} > f_{10}) \wedge (f_{00} > f_{11})\} + \\
&\sum_{(y_1,y_2,y_3)} f_{01} : \{(f_{01} > f_{00}) \wedge (f_{01} > f_{10}) \wedge (f_{01} > f_{11}))\} + \\
&\sum_{(y_1,y_2,y_3)} f_{10} : \{(f_{10} > f_{00}) \wedge (f_{10} > f_{01}) \wedge (f_{10} > f_{11})\} + \\
&\sum_{(y_1,y_2,y_3)} f_{11} : \{(f_{11} > f_{00}) \wedge (f_{11} > f_{01}) \wedge (f_{11} > f_{10})\} + \\
&\sum_{(y_1,y_2,y_3)} f_{00} : \{(f_{00} = f_{01}) \wedge (f_{00} \neq f_{10}) \wedge (f_{00} \neq f_{11})\} +
\end{aligned}
$$

$$\sum_{(y_1,y_2,y_3)} f_{00} : \{(f_{00} = f_{01}) \wedge (f_{00} \neq f_{10}) \wedge (f_{00} \neq f_{11})\} +$$

$$\sum_{(y_1,y_2,y_3)} f_{00} : \{(f_{00} = f_{11}) \wedge (f_{00} \neq f_{01}) \wedge (f_{00} \neq f_{10})\} +$$

$$\sum_{(y_1,y_2,y_3)} f_{01} : \{(f_{01} = f_{10}) \wedge (f_{01} \neq f_{00}) \wedge (f_{01} \neq f_{11}) +$$

$$\sum_{(y_1,y_2,y_3)} f_{01} : \{(f_{01} = f_{11}) \wedge (f_{01} \neq f_{00}) \wedge (f_{01} \neq f_{10})\} +$$

$$\sum_{(y_1,y_2,y_3)} f_{10} : \{(f_{10} = f_{11}) \wedge (f_{10} \neq f_{00}) \wedge (f_{10} \neq f_{01})\} +$$

$$\sum_{(y_1,y_2,y_3)} f_{00} : \{(f_{00} = f_{10} = f_{11}) \wedge (f_{00} \neq f_{01})\} +$$

$$\sum_{(y_1,y_2,y_3)} f_{00} : \{(f_{00} = f_{01} = f_{10}) \wedge (f_{00} \neq f_{11})\} +$$

$$\sum_{(y_1,y_2,y_3)} f_{00} : \{(f_{00} = f_{01} = f_{11}) \wedge (f_{00} \neq f_{10})\} +$$

$$\sum_{(y_1,y_2,y_3)} f_{01} : \{(f_{01} = f_{10} = f_{11}) \wedge (f_{01} \neq f_{00})\} +$$

$$\sum_{(y_1,y_2,y_3)} f_{00} : \{(f_{00} = f_{01} = f_{10} = f_{11})\}.$$

Where

$$f_{00} \equiv (1-p)^2 \cdot \text{Poiss}(Y_1; \lambda_0 T_1) \cdot \text{Poiss}(Y_2; \lambda_0 T_2) \cdot \text{Poiss}(Y_3; 2\lambda_0 T_3),$$

$$f_{01} \equiv p(1-p) \cdot \text{Poiss}(Y_1; \lambda_0 T_1) \cdot \text{Poiss}(Y_2; \lambda_1 T_2) \cdot \text{Poiss}(Y_3; (\lambda_0 + \lambda_1) T_3),$$

$$f_{10} \equiv p(1-p) \cdot \text{Poiss}(Y_1; \lambda_1 T_1) \cdot \text{Poiss}(Y_2; \lambda_0 T_2) \cdot \text{Poiss}(Y_3; (\lambda_1 + \lambda_0) T_3),$$

$$f_{11} \equiv p^2 \cdot \text{Poiss}(Y_1; \lambda_1 T_1) \cdot \text{Poiss}(Y_2; \lambda_1 T_2) \cdot \text{Poiss}(Y_3; 2\lambda_1 T_3).$$

Appendix C

Numerical Approximation of the Poisson pmf and Mutual Information

$$\mathrm{Poiss}(x;\lambda) \approx \begin{cases} \frac{e^{\lambda}\lambda^{x}}{x!} & x \leq x_c \\ 0 & x > x_c \end{cases} \tag{C.1}$$

where x_c is such that cummulative distribution function of Poisson random variable at x_c, $\mathrm{CDFPoiss}(x_c;\lambda)$, is equal to 1 represented in double precision floating-point arithmetic in IEEE 754 standard. This means that numerical values of $\mathrm{CDFPoiss}(x_c;\lambda)$ and $\mathrm{CDFPoiss}(x_c+1;\lambda)$ are identical and equal to one.

The infinite summations in mutual information expression given in Appendix A are

all truncated from 0 to $x_c = \mathrm{CDFPoiss}^{-1}(1-\epsilon;(\lambda_1+\lambda_1)T)$ for a given value of λ_1 and T. $\mathrm{CDFPoiss}^{-1}$ is inverse Poisson cdf and ϵ is, the machine precision number, 2^{-53} [21].

Appendix D

Code description

Since all computations are independent of any other computations, we may exploit this fact by vectorization of the algorithm and then evaulating MI at grid points in parallel as defined in flowchart in Fig.(D.1) and Fig.(D.2) for fast computational throughput. However, it must be noted that this way of vectorized computations by first performing the truncation of Poisson distribution and then forming the grid of points, for further calculation of MI, is not effective for large values of Poisson intensities involved in the problem. This is because the grid size increases cubely w.r.t high end of the truncation. So, for large values of Poisson intensities, say $\lambda >> 100$, we need to resort to Monte Carlo method.

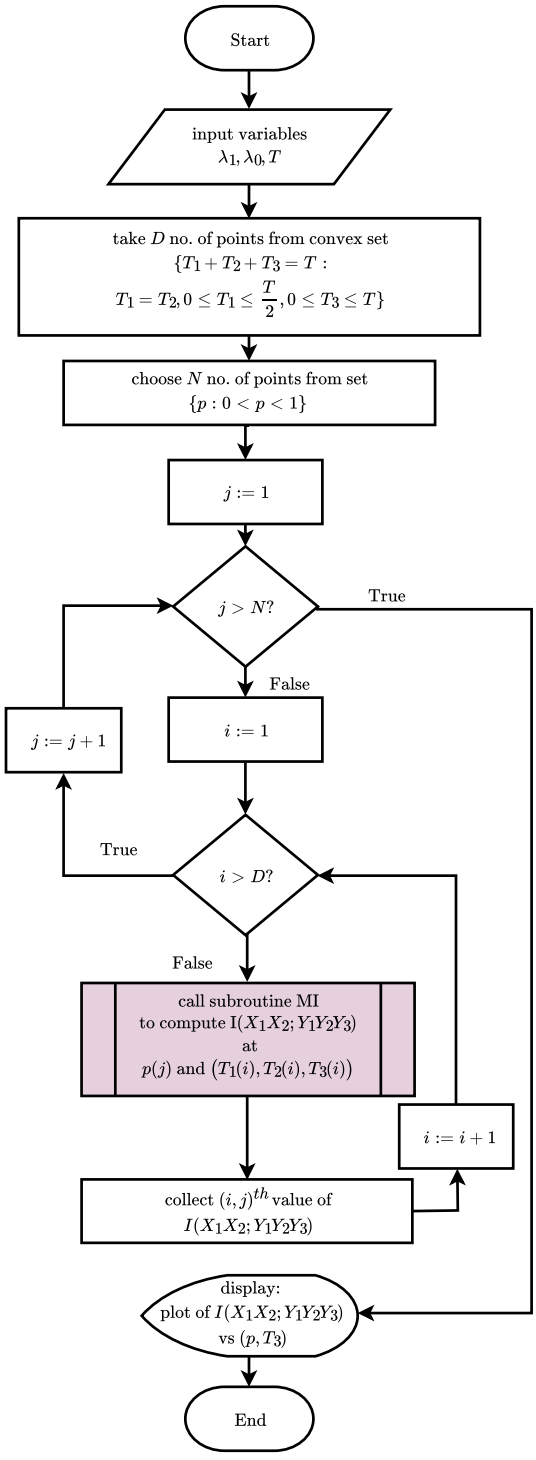

Figure D.1: Flowchart 1: Algorithm description for full support of targets detection.

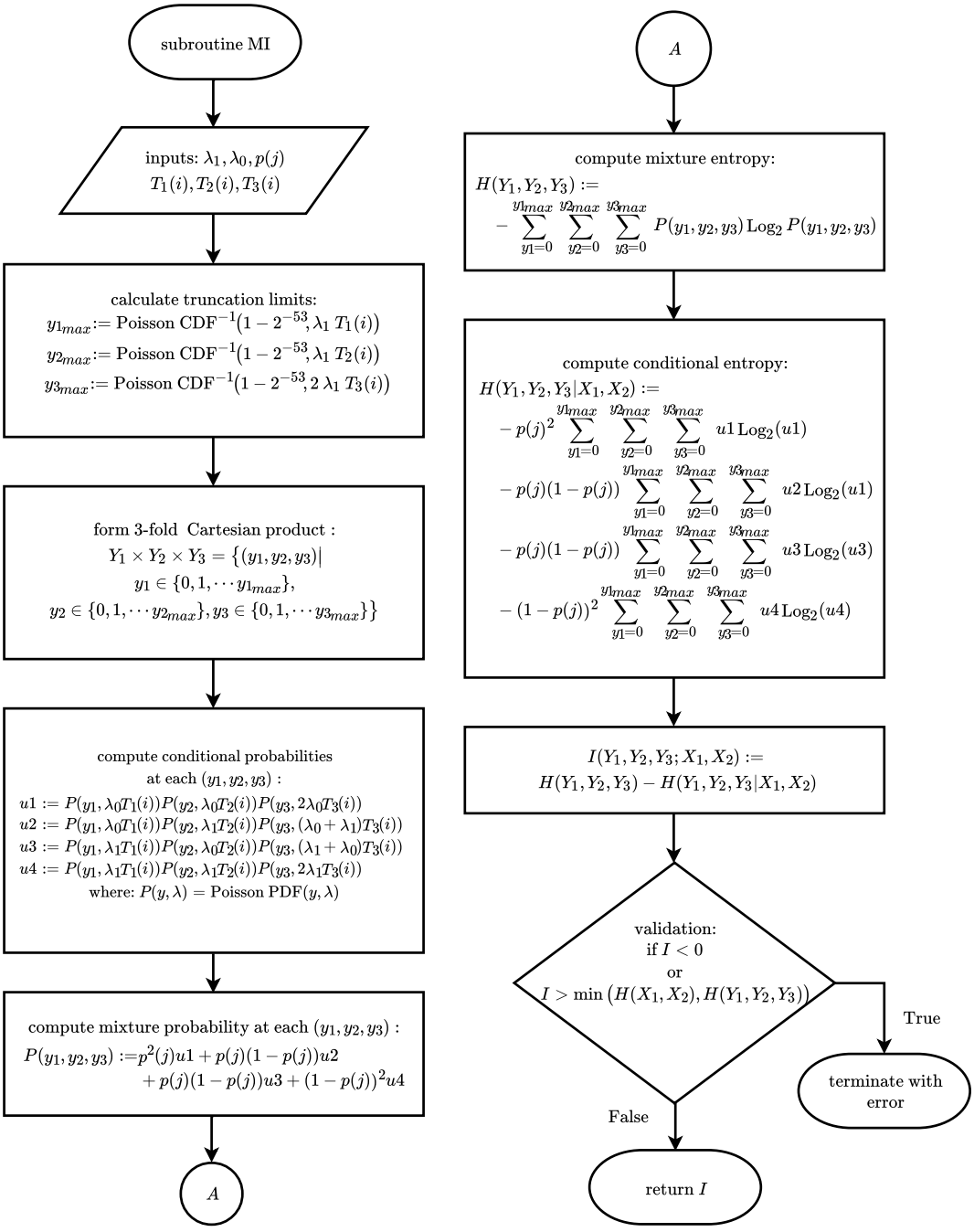

Figure D.2: Flowchart 2: subroutine (two target detection problem).

Appendix E

Sharp bounds on the Poisson entropy

The asymptotically tight bounds for Poisson entropy (for small and large intensities) given in [1] helps in validation of our computed entropy of Poisson variable (which is computed by truncation of Poisson PDF) as defined in flowcharts (however, bounds for the entropy of multivariate Poisson mixture remains unanswered). Since bound gap (for small λ) differ by $\mathcal{O}(\lambda^{2m+1})$ for given m, we can achieve any accuracy by setting the value of m but at a higher computational cost. For large λ the bound gap decreases by $\mathcal{O}(\lambda^{-m})$ w.r.t λ for a fixed m. We checked our computed entropies in our interested ranges of λ for up to $m = 6$ and got the values within bounds. For large values of λ when $m = 4$ as given in Fig. (E.1):

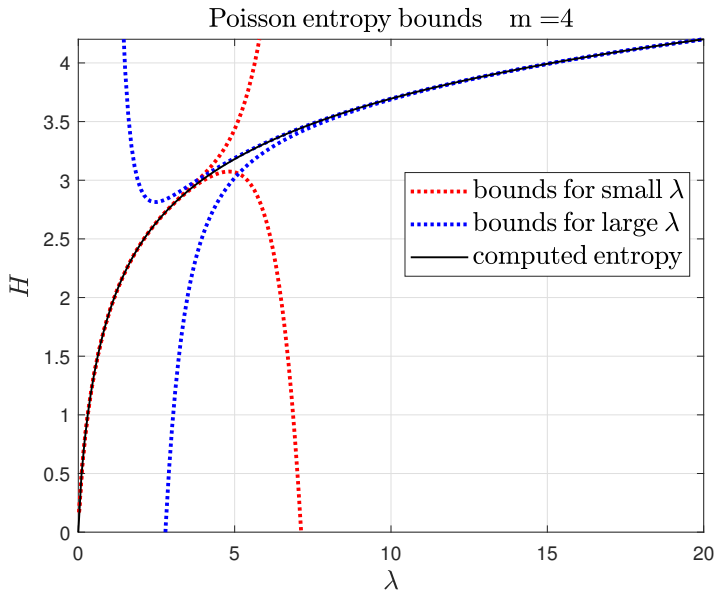

Figure E.1: Bounds for small values of λ (red curves) are from theorem 1 of [1]. Bounds for large values of λ (blue curves) are results of theorem 2. (MATLAB-CODE F.17 & F.18.3)

Coefficients $b(m,k)$ for upper bound: $\frac{1}{504\lambda^7}+\frac{571}{1008\lambda^6}+\frac{12367}{2520\lambda^5}+\frac{201}{80\lambda^4}$
$-\frac{19}{360\lambda^3}-\frac{1}{24\lambda^2}-\frac{1}{12\lambda}.$

Coefficients $a(m,k)$ for lower bound: $\frac{1}{72\lambda^8}+\frac{167}{21\lambda^7}+\frac{2275}{18\lambda^6}+\frac{210}{\lambda^5}+\frac{105}{4\lambda^4}.$

Above coefficients can be verified from Fig.(1) of [1] provided in the last column.

Appendix F

Matlab Codes of Figures

F.1 Code for Fig. (2.2)

```
clear all
close all

p=0.5;
i=1;
D=400;
t=10;
TT=[  ];
pd= [ ];
% for T=0:0.25:t;
for T=linspace(0,t,D);
```

```
L0=1*T;
L1=5*T;

x0_min= poissinv(eps(0.5),L0);
x0_max= poissinv(1-eps(0.5),L0);

%%%%%%%%%%%%%%%%%%%%%%%%%%
x_min= poissinv(eps(0.5),L1);
x_max= poissinv(1-eps(0.5),L1);

%%%%%%%%%%%%%%
x1=x0_min;
x2=x_max;
py=(1-p)*poisspdf(x1:x2,L0) +p*poisspdf(x1:x2,L1);
% py=(1-p)*poisspdf(0:x2,L0) +p*poisspdf(0:x2,L1);
l_py=log2(py);
dy= - py  .*  l_py;

Hy=nansum(dy);

pyx0=poisspdf(x1:x2,L0);
pyx1=poisspdf(x1:x2,L1);

% pyx0=poisspdf(0:x2,L0);
% pyx1=poisspdf(0:x2,L1);

% HHyx=(1-p)*sum(-pyx0.*log2(pyx0)) + p*sum(-pyx1.*log2(↩
   pyx1))

z0= - pyx0  .*  log2(pyx0);
% z0(isnan(z0))=0;   % converts NaN to 0.
% Z0=sum(z0);  % Entropy of y

Z0=nansum(z0);
```

```
z1= - pyx1   .*  log2(pyx1);
% z1(isnan(z1))=0;   % converts NaN to 0.
% Z1=sum(z1); % Entropy of y

Z1=nansum(z1);

Hyx=(1-p)*Z0 + p*Z1;  % H(Y|X)

I(i)=Hy-Hyx;

%%%%%%%%%%%%%%%%%%%
% [u,v]=meshgrid(0:1,x1:x2);
z=[(1-p)*poisspdf(x1:x2,L0)'  p*poisspdf(x1:x2,L1)'];

lz=-log2(z);
h=z.*lz;

H_xy=nansum(nansum(h));

info(i)=1+Hy-H_xy; % incorrect
pd=cat(2,pd,sum(max(z,[ ],2)));
%%%%%%%%%%%%%%%%%%%

i=i+1;
TT=cat(2,TT,T);
end

H=-(p*log2(p)+(1-p)*log2(1-p))  % Prior Entropy
%  figure('Units','inches','Position',[0 0 4 2],'↩
   PaperPositionMode','auto')
figure;
yyaxis left
plot([0 t],[H H],':k',TT,I,'b','LineWidth',1.5)
```

```
ylabel('$I(X_1;Y_1)$','FontUnits','points','FontSize'↩
   ,14,'interpreter','latex');

hold on
yyaxis right
plot(TT,pd,'r','LineWidth',1.5)
ylabel('$P_d$','FontUnits','points','FontSize',14,'↩
   interpreter','latex');

set(gca,'FontSize',14)

xlabel('$T$','FontUnits','points','FontSize',14,'↩
   interpreter','latex');
title(['$p= $' num2str(p)...
'$\quad \lambda_0= $' num2str(L0/t)...
'$\quad  \lambda_1=$' num2str(L1/t)],'FontUnits','points↩
   ','FontSize',14,'Interpreter','latex');

h=legend('$H(X_1)$','$I$','$P_d$')
set(h,'FontUnits','points','FontSize',14,'Interpreter','↩
   latex')
grid on
```

F.2 Code for Fig. (2.3). (Include code F.18.1)

```
% Probability of Detection(analytical), Correct Decision↩
    Rate(Empirical)
% and Mutual Information versus time T3, for Full Target↩
    Support
```

```
clear all
close all

tic
p=0.1;     % Probability of 1 (higher rate)
T=1;      % Total time
L0=5;    L1=20;     % Poisson rates
q=1-p;

% s=10^4;  % total number of samples / Data point.

global X
X=1;

D=100;  % Total number of data points.
v=[linspace(0,T/2,D)',linspace(0,T/2,D)',linspace(T,0,D)↩
   ']; % samples of (T1,T2,T3)

MI=[ ];
% Cd=[ ];
% Pd=[ ];

for i=1:length(v);
[m1,m2,m3,m,hy]=Con_MI_2(p,L0,L1,v(i,1),v(i,2),v(i,3)); ↩
    % Evaluate mutual information for each point (T1,T2,↩
   T3).
MI=cat(2,MI,[m1;m2;m3;m;hy]); % I1,    I2,     I3,      I, ↩
      I1+I2+I3,       Hy
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
mi=MI(4,:);
[f1,g1]=max(mi);  % maximum value of MI
v(g1,:)
figure;
```

```
ft=14; %font size
plot(v(:,3),MI(1,:),'g','LineWidth',2);
set(gca,'FontSize',ft)
hold on; plot(v(:,3),MI(2,:),'--r','LineWidth',2);
hold on; plot(v(:,3),MI(3,:),'--b','LineWidth',2);
hold on; plot(v(:,3),MI(1,:)+MI(2,:)+MI(3,:),'c','↩
   LineWidth',2);
hold on; plot(v(:,3),MI(4,:),'--k','LineWidth',2);
xlabel('$T_3$','FontUnits','points','FontSize',ft,'↩
   Interpreter','latex');
ylabel('$I$','FontUnits','points','FontSize',ft,'↩
   Interpreter','latex');
title({['$I(X_1 X_2;Y_1 Y_2 Y_3)=I(X_1;Y_1)+I(X_2;Y_2)+I↩
   (X_1 X_2;Y_3|Y_1  Y_2).$'],['$\scriptstyle{\mathop {\↩
   arg \max }\limits_{T3} I} = $'...
num2str(v(g1,3))  '$\quad p= $' num2str(p)...
'$\quad \lambda_0= $' num2str(L0)...
'$\quad  \lambda_1=$' num2str(L1) ...
'$\quad T=$' num2str(T)]},'FontUnits','points','FontSize↩
   ',ft,'Interpreter','latex');

h=legend('$I_1: I(X_1;Y_1)$','$I_2: I(X_2;Y_2)$','$I_3: ↩
   I(X_1 X_2;Y_3|Y_1  Y_2)$','$I_1+I_2+I_3$','$I \mbox{(↩
   brute force calculated)}$','Location','west');
set(h,'FontUnits','points','FontSize',ft,'Interpreter','↩
   latex');
grid on
%%
figure
E=[q^2  p*q p*q p^2]; Hh=-E*log2(E');
plot(v(:,3)',Hh*ones(1,length(v(:,3))),'b','LineWidth'↩
   ,2); set(gca,'FontSize',ft);
xlabel('$T_3$','FontUnits','points','FontSize',ft,'↩
   Interpreter','latex');
```

```
ylabel('$H$','FontUnits','points','FontSize',ft,'↩
   Interpreter','latex');
hold on; plot(v(:,3),MI(5,:),'r','LineWidth',2);
title({['$\scriptstyle{\mathop {\arg \max }\limits_{T3} ↩
   MI} = $'...
num2str(v(g1,3))  '$\quad p= $' num2str(p)...
'$\quad \lambda_0= $' num2str(L0)...
'$\quad  \lambda_1=$' num2str(L1) ...
'$\quad T=$' num2str(T)]},'FontUnits','points','FontSize↩
   ',ft,'Interpreter','latex');
h=legend('$H(X)$','$H(Y)$');
set(h,'FontUnits','points','FontSize',ft,'Interpreter','↩
   latex');
grid on
toc
```

F.3 Code for Fig. (2.4). (Include code F.18.2)

```
% Probability of Detection(analytical), Correct Decision↩
    Rate(Empirical)
% and Mutual Information versus time T3, for Full Target↩
    Support

clear all
close all

tic
% s=10^4;  % total number of samples / Data point.

global X
```

```
X=1;

T=15;      % Total time
D=100;  % Total number of data points.

v=[linspace(0,T/2,D)',linspace(0,T/2,D)',linspace(T,0,D)↩
   ']; % samples of (T1,T2,T3)

L0=2;   L1=5;     % Poisson rates

% P1=linspace(eps(0),2^(-8),10) ;
% P2=linspace(2^(-8)+eps(2^-8),2^(-4),40) ;
% P3=linspace(2^(-4)+eps(2^-4),2^(-2),40) ;
% P4=linspace(2^(-2)+eps(2^-2),2^(-1),50) ;
% P5=linspace(2^(-1)+eps(2^-1),0.99999,50) ;
% P6=linspace(0.999999+eps(0.999999),1-5*eps(0.5),10) ;
% P=[P1 P2 P3 P4 P5 P6];

P1=linspace(eps(0),2^(-3),10) ;
P2=linspace(2^(-3)+eps(2^-3),2^(-2),10) ;
P3=linspace(2^(-2)+eps(2^-2),3*2^(-2),30) ;
P4=linspace(3*2^(-2)+eps(3*2^-2),7*2^(-3),10) ;
P5=linspace(7*2^(-3)+eps(7*2^-3),0.99999,10) ;

P=[P1 P2 P3 P4 P5];

N=length(P); %Number of probability points
% N=200; %Number of probability points
% P=linspace(0+eps,0.005,N);
% P=linspace(1-10*eps,1-eps,N);
% P=linspace(.995,1,N);
% P=1-eps(0.5);

O_MI=[ ];
```

```
O_T3=[ ];
O_p= [ ];
m_info=[ ];

for j=1:N;
p=P(j);    % Probability of 1 (higher rate)
% q=1-p;

MI=[ ];
% Cd=[ ];
% Pd=[ ];

for i=1:length(v);
[m,h0,h1,h2,h3]=MI_2(p,L0,L1,v(i,1),v(i,2),v(i,3));  % ↩
   Evaluate mutual information for each point (T1,T2,T3).
%[m,h0,h1,h2,h3]=My_MI(p,L0,L1,v(i,1),v(i,2),v(i,3));  %↩
    Evaluate mutual information for each point (T1,T2,T3)↩
   .
MI=cat(2,MI,m);

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[f1,g1]=max(MI);  % maximum value of MI
% v(g1,:)
m_info=cat(1,m_info,MI); % size: length(P) X D
O_MI=cat(2,O_MI,f1);   % Optimal MI  size: 1 X length(P)
O_T3=cat(2,O_T3,v(g1,3));    % Optimal T3 size: 1 X ↩
   length(P)
O_p=cat(2,O_p,p);    % Value of p for that particular ↩
   optimal MI and optimal T3
%%%%%%%%%
%%%%%%%%%
hold on
plot(v(:,3)',MI, 'LineWidth',2);
```

```
% figure; plot(O_p(1:190),O_T3(1:190), 'LineWidth',2);
ylabel('$MI$','Interpreter','latex');
xlabel('$T_3^0$','Interpreter','latex');

title(['$ P(\lambda_1)=p$' blanks(1)...
'$\quad \lambda_0= $' num2str(L0)...
'$\quad  \lambda_1=$' num2str(L1) ...
'$\quad T=$' num2str(T) ],'Interpreter','latex');
% keyboard
%%%%%%%
%%%%%%%
end


figure; plot(O_p,O_T3, 'LineWidth',2);
% figure; plot(O_p(1:190),O_T3(1:190), 'LineWidth',2);
xlabel('$p$','Interpreter','latex');
ylabel('$T_3^0$','Interpreter','latex');

title(['$ P(\lambda_1)=p$' blanks(1)...
'$\quad \lambda_0= $' num2str(L0)...
'$\quad  \lambda_1=$' num2str(L1) ...
'$\quad T=$' num2str(T) ],'Interpreter','latex');

figure; plot(O_p,O_MI, 'LineWidth',2);
xlabel('$p$','Interpreter','latex');
ylabel('$MI^{O}$','Interpreter','latex');
title(['$ P(\lambda_1)=p$' blanks(1)...
'$\quad \lambda_0= $' num2str(L0)...
'$\quad  \lambda_1=$' num2str(L1) ...
'$\quad T=$' num2str(T) ],'Interpreter','latex');

figure;stem(O_p,ones(1,length(O_p)),'Marker', 'none'); ↩
   xlabel('$p$','Interpreter','latex');
```

```
title(['No of $p$ samples=' num2str(length(O_p)) '\quad ↩
   No of divisons of $T $=' num2str(length(v)) '$\quad  P↩
   (\lambda_1)=p$' blanks(1)...
'$\quad \lambda_0= $' num2str(L0)...
'$\quad  \lambda_1=$' num2str(L1) ...
'$\quad T=$' num2str(T) ],'Interpreter','latex');

p_info=repmat(P',[1,D]);
B=flipud(v(:,3)');
% T3_info=repmat(v(:,3)',[length(P),1]);
T3_info=repmat(B,[length(P),1]);
% figure; surf(T3_info,p_info,m_info)
figure; mesh(p_info,T3_info,m_info)
ylabel('$p$','Interpreter','latex');
xlabel('$T_3$','Interpreter','latex');

title(['$ P(\lambda_1)=p$' blanks(1)...
'$\quad \lambda_0= $' num2str(L0)...
'$\quad  \lambda_1=$' num2str(L1) ...
'$\quad T=$' num2str(T) ],'Interpreter','latex');

figure;contourf(T3_info,p_info,m_info)
toc
```

F.4 Code for Fig. (2.5)

```
clear all
close all

L0=2;
```

```
L1=100;

p=linspace(0.00001,0.99999,200);

D1=(1-p).^2*(2*L0)*log2(2*L0)+2*(1-p).*p*(L0+L1)*log2(L0+L1)+p.^2*(2*L1)*log2(2*L1)-...
    ((1-p).^2*2*L0+2*(1-p).*p*(L0+L1)+p.^2*2*L1).*log2( (1-p).^2*2*L0+2*(1-p).*p*(L0+L1)+p.^2*2*L1);

 D2= (1-p)*(L0)*log2(L0)+p*(L1)*log2(L1)-((1-p).*L0+p.*L1).*log2((1-p).*L0+p.*L1);

figure; plot(p,[D1], 'LineWidth',2);
axis tight
grid on
xlabel('$p$','FontSize',14,'Interpreter','latex');
ylabel('$\frac{d}{dT} I |_{T=0}$','FontSize',14,'Interpreter','latex');
title(['$ P(\lambda_1)=p$' blanks(1)...
'$\quad \lambda_0= $' num2str(L0)...
'$\quad  \lambda_1=$' num2str(L1) ],'FontSize',14,'Interpreter','latex');
h=legend('$\frac{d}{dT} I(U;Y_3) \Big|_{T=0}$','$\frac{d}{dT} I(U;Y_1,Y_2) \Big|_{T=0}$');
set(h,'Location','southeast','FontUnits','points','FontSize',14,'Interpreter','latex','location','best');

figure; plot(p,[D1;D2], 'LineWidth',2);
axis tight
grid on
xlabel('$p$','FontSize',14,'Interpreter','latex');
ylabel('$\frac{d}{dT} I |_{T=0}$','FontSize',14,'Interpreter','latex');
title(['$ P(\lambda_1)=p$' blanks(1)...
```

```
'$\quad \lambda_0= $' num2str(L0)...
'$\quad  \lambda_1=$' num2str(L1) ],'FontSize',14,'↩
   Interpreter','latex');
h=legend('$\frac{d}{dT} I(X_1,X_2;Y_3) \Big|_{T=0}$','$\↩
   frac{d}{dT} I(X_1,X_2;Y_1,Y_2) \Big|_{T=0}$');
set(h,'Location','southeast','FontUnits','points','↩
   FontSize',14,'Interpreter','latex','location','best');
```

F.5 Code for Fig. (2.6)

```
%% Main program (Binomial input for general N, in scalar↩
    Poisson)
clear all
close all

tic
global X
X=1;
N=2;
p=0.25;  q=1-p;
L0=100;     L1=200;
% L0=2;     L1=3;
% e=eps(0.5);
d=30; %Odd number to ensure mid point of vector t to ↩
   include
t0=1; %total time
t=[0 0.000001 linspace(0.001,t0,d)];  % Total time
% t=[0 eps(0) linspace(eps(0)+eps(eps(0)),t0,d)];  % ↩
   Total time
MI=[ ];
```

```
H_u=[ ];
H_ux=[ ];
hc=[ ];
% MI_u=[ ];
% MI2=[ ];
b=[ ]; % Binomial coeficients initialization
I_u=[ ];
g=[ ];
% for r=0:N;
%     bb=nchoosek(N,r);
%     b=cat(2,b,bb); % Binomial coeficients
%     f=q^(N-r)*p^(r);
%     g=cat(2,g,f);
% end
% Bp= binopdf(0:N,N,p);
%  C=Bp.*b;
%  FF=b.*g;
% C=Bp;
FF= binopdf(0:N,N,p);  % Binomial distrubtuion B(N=10,p)
for i=1:length(t)
X
T=t(i);

tt=T/N; %scaling
x0_max= poissinv(1-eps(0.5),L0*tt);
x0_min= poissinv(eps(0.5),L0*tt);

x1_max= poissinv(1-eps(0.5),L1*tt);
x1_min= poissinv(eps(0.5),L1*tt);

py0=poisspdf(x0_min:x0_max,L0*tt);
py1=poisspdf(x1_min:x1_max,L1*tt);
```

```
py=q*poisspdf(min(x0_min,x1_min):max(x0_max,x1_max),L0*↩
   tt)+...
    p*poisspdf(min(x0_min,x1_min):max(x0_max,x1_max),L1*↩
       tt); %max and min are introduced
% to prevent numerical artifact
H_y=-nansum(py.*log2(py));
h0=-q*nansum(py0.*log2(py0));
h1=-p*nansum(py1.*log2(py1));

m=H_y-(h0+h1);
MI=cat(2,MI,m);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
m_u=[ ];
    L=N*max([L0 L1])*T;
%     x_min= poissinv(eps(0.5),L);
    x_max=poissinv(1-eps(0.5),L);
for k=0:N   %Mixture probability
    pp=poisspdf(0:x_max,((N-k)*L0+k*L1)*T);
    pu=FF(k+1)*pp;
    m_u=cat(1,m_u,pu);
end
ku=nansum(m_u,1);
hu=-nansum(nansum(ku.*log2(ku)));
H_u=cat(2,H_u,hu);
hux=0;
for w=0:N   % For conditional probabilities
    x_min= poissinv(eps(0.5),((N-w)*L0+w*L1)*T);
    x_max=poissinv(1-eps(0.5),((N-w)*L0+w*L1)*T);
    p1=poisspdf(x_min:x_max,((N-w)*L0+w*L1)*T);
    hux=hux-FF(w+1)*p1*log2(p1)';
end
% keyboard
```

```
% H=nansum(nansum(m_u)) %entropy of multivariate
   poisson mixture
% H_u=cat(2,H_u,H);
H_ux=cat(2,H_ux,hux);
I=hu-hux;
I_u=cat(2,I_u,I);
X=X+1;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% %% fig 1
% figure
% plot(2*t(1:ceil(d/2)),N*MI(1:ceil(d/2)),'LineWidth',2)
% hold on
% plot(2*[t(1) t(ceil(d/2))],N*[-(q*log2(q)+p*log2(p))
   ,-(q*log2(q)+p*log2(p))],'LineWidth',2)
% xlabel('$T$ (scaled by $2$)','FontUnits','points','
   FontSize',14,'interpreter','latex');
% ylabel('$2 \cdot I_1(X_1;Y_1)$','FontUnits','points','
   FontSize',14,'interpreter','latex');
% title(['$p= $' num2str(p)...
% '$\quad \lambda_0= $' num2str(L0)...
% '$\quad  \lambda_1=$' num2str(L1)],'FontUnits','points
   ','FontSize',14,'Interpreter','latex');
% h=legend('$2 \cdot I_1 |_{T:=\frac{T}{2}}$');
% set(h,'FontUnits','points','FontSize',14,'Interpreter
   ','latex');

%% fig 2
figure
% plot(t,I_u,'LineWidth',2)
plot(t,I_u,'LineWidth',2)
hold on
y=-(binopdf(0:N,N,p))*(log2(binopdf(0:N,N,p)))';
```

```
plot([t(1) t(end)],[y y],'LineWidth',2)
xlabel('$T$','FontUnits','points','FontSize',14,'↩
   interpreter','latex');
ylabel('$ I(U;Y_3)$','FontUnits','points','FontSize',14,↩
   'interpreter','latex');
title(['$N= $' num2str(N) '$\quad p= $' num2str(p)...
'$\quad \lambda_0= $' num2str(L0)...
'$\quad  \lambda_1=$' num2str(L1)],'FontUnits','points',↩
   'FontSize',14,'Interpreter','latex');
h=legend('$I(U;Y_3)$','$H(U)$');
set(h,'FontUnits','points','FontSize',14,'Interpreter','↩
   latex');

% %% fig 3
% figure
% plot(2*t(1:ceil(d/2)),N*MI(1:ceil(d/2)),'LineWidth',2)
% hold on
% plot(2*[t(1) t(ceil(d/2))],N*[-(q*log2(q)+p*log2(p)) ↩
   ,-(q*log2(q)+p*log2(p))],'LineWidth',2)
% xlabel('$T$','FontUnits','points','FontSize',14,'↩
   interpreter','latex');
% ylabel('$N \cdot I_1(X_1;Y_1)$','FontUnits','points','↩
   FontSize',14,'interpreter','latex');
% title(['$N= $' num2str(N) '$\quad p= $' num2str(p)...
% '$\quad \lambda_0= $' num2str(L0)...
% '$\quad  \lambda_1=$' num2str(L1)],'FontUnits','points↩
   ','FontSize',14,'Interpreter','latex');
% h=legend('$N \cdot I_1 |_{T:=\frac{T}{2}}$','$N \cdot ↩
   H(X_1)$');
% set(h,'FontUnits','points','FontSize',14,'Interpreter↩
   ','latex');

%% fig 3
figure
```

```
plot(t,N*MI,'LineWidth',2);
hold on
plot([t(1) t(end)],N*[-(q*log2(q)+p*log2(p)) ,-(q*log2(q↩
   )+p*log2(p))],'LineWidth',2)
xlabel('$T$','FontUnits','points','FontSize',14,'↩
   interpreter','latex');
% ylabel('$N \cdot I_1(X_1;Y_1)$','FontUnits','points','↩
   FontSize',14,'interpreter','latex');
ylabel('$I$','FontUnits','points','FontSize',14,'↩
   interpreter','latex');
title(['$N= $' num2str(N) '$\quad p= $' num2str(p)...
'$\quad \lambda_0= $' num2str(L0)...
'$\quad  \lambda_1=$' num2str(L1)],'FontUnits','points',↩
   'FontSize',14,'Interpreter','latex');
% h=legend('$N \cdot I_1 |_{T:=\frac{T}{2}}$','$N \cdot ↩
   H(X_1)$');
h=legend('$I(X_1, X_2 \cdots X_N;Y_1, Y_2, \cdots Y_N)$'↩
   ,'$H(X_1,X_2 \cdots X_N)$');
set(h,'Location','southeast','FontUnits','points','↩
   FontSize',14,'Interpreter','latex');


%% fig 4
figure
plot(t,[N*MI;I_u],'LineWidth',2);
% hold on
% plot([t(1) t(end)],N*[-(q*log2(q)+p*log2(p)) ,-(q*log2↩
   (q)+p*log2(p))],'LineWidth',2)
xlabel('$T$','FontUnits','points','FontSize',14,'↩
   interpreter','latex');
% ylabel('$N \cdot I_1(X_1;Y_1)$','FontUnits','points','↩
   FontSize',14,'interpreter','latex');
ylabel('$I$','FontUnits','points','FontSize',14,'↩
   interpreter','latex');
```

```
title(['$N= $' num2str(N) '$\quad p= $' num2str(p)...
'$\quad \lambda_0= $' num2str(L0)...
'$\quad  \lambda_1=$' num2str(L1) '$\quad  H(\sum_i X_i)↩
   =$' num2str(-FF*log2(FF')),...
'$\quad  H( X_1 \cdots X_N)=$' num2str(-N*(q*log2(q)+p*↩
   log2(p)) ) ],'FontUnits','points','FontSize',14,'↩
   Interpreter','latex');
h=legend('$I(X_1, X_2 \cdots X_N;Y_1, Y_2, \cdots Y_N)$'↩
   ,...
'$I(\sum_i X_i;Y_{2^N-1})$');
set(h,'Location','southeast','FontUnits','points','↩
   FontSize',14,'Interpreter','latex');


%% fig 5
figure
plot(t,[N*MI;I_u],'LineWidth',2);
% hold on
% plot([t(1) t(end)],N*[-(q*log2(q)+p*log2(p)) ,-(q*log2↩
   (q)+p*log2(p))],'LineWidth',2)
xlabel('$T$','FontUnits','points','FontSize',14,'↩
   interpreter','latex');
% ylabel('$N \cdot I_1(X_1;Y_1)$','FontUnits','points','↩
   FontSize',14,'interpreter','latex');
ylabel('$I$','FontUnits','points','FontSize',14,'↩
   interpreter','latex');
title(['$N= $' num2str(N) '$\quad p= $' num2str(p)...
'$\quad \lambda_0= $' num2str(L0)...
'$\quad  \lambda_1=$' num2str(L1) '$\quad  H(\sum_i X_i)↩
   =$' num2str(-FF*log2(FF')),...
'$\quad  H( X_1 \cdots X_N)=$' num2str(-N*(q*log2(q)+p*↩
   log2(p)) ) ],'FontUnits','points','FontSize',14,'↩
   Interpreter','latex');
```

```
%% fig 5
% figure
% semilogx(t,[N*MI;I_u],'LineWidth',2);
% xlabel('$T$','FontUnits','points','FontSize',14,'↩
   interpreter','latex');
% ylabel('$I$','FontUnits','points','FontSize',14,'↩
   interpreter','latex');
% title(['$N= $' num2str(N) '$\quad p= $' num2str(p)...
% '$\quad \lambda_0= $' num2str(L0)...
% '$\quad  \lambda_1=$' num2str(L1) '$\quad  H(\sum_i ↩
   X_i)=$' num2str(-FF*log2(FF')),...
% '$\quad  H( X_1 \cdots X_N)=$' num2str(-N*(q*log2(q)+p↩
   *log2(p)) ) ],'FontUnits','points','FontSize',14,'↩
   Interpreter','latex');
% h=legend('$I(X_1, X_2 \cdots X_N;Y_1, Y_2, \cdots Y_N)↩
   $',...
% '$I(\sum_i X_i;Y_{2^N-1})$');
% set(h,'Location','southeast','FontUnits','points','↩
   FontSize',14,'Interpreter','latex');

%% Numerical dervative at T=0
yy=N*MI;
d1=(yy(2)-yy(1))/(t(2)-t(1)) % derivative at T=0 for T1=↩
   T2=T/2
d2=(I_u(2)-I_u(1))/(t(2)-t(1)) % derivative at T=0 for ↩
   T3=T

for s=0:N
    n(s+1)=L0*(N-s)+L1*(s);
end

%% Analytic derivatives
e3=(1-p)*(L0*log2(L0))+p*(L1*log2(L1));
e4=(1-p)*(L0)+p*(L1);
```

```
e5=e4*log2(e4);
dd1=e3-e5 %% Analytic derivative at T=0: For T1=T2=T/2 ↩
   case
%%%%%%%%%%%%%%%
e1=nansum(FF.*(n.*log2(n)));
e2=nansum(FF.*(n))*log2(nansum(FF.*(n)));
dd2=e1-e2 %% Analytic derivative at T=0: For T=T3 case

hold on
x = [t(1) t(end)];
y = [yy(1) dd1*t(end)];
line(x,y,'Color','blue','LineStyle','--')

hold on
x = [t(1) t(end)];
y = [I_u(1) dd2*t(end)];
line(x,y,'Color','red','LineStyle','--')
axis tight

h=legend('$I(X_1, X_2 \cdots X_N;Y_1, Y_2, \cdots Y_N)$'↩
   ,...
'$I(\sum_i X_i;Y_{2^N-1})$','$\textrm{Tangent to blue ↩
   curve at } T=0$','$\textrm{Tangent to red curve at } T↩
   =0 $');
set(h,'Location','southeast','FontUnits','points','↩
   FontSize',14,'Interpreter','latex','location','best');
grid on

%%
%% fig 6
figure
plot(t,[N*MI;I_u],'LineWidth',2);
% hold on
```

```
% plot([t(1) t(end)],N*[-(q*log2(q)+p*log2(p)) ,-(q*log2(q)+p*log2(p))],'LineWidth',2)
xlabel('$T$','FontUnits','points','FontSize',14,'interpreter','latex');
% ylabel('$N \cdot I_1(X_1;Y_1)$','FontUnits','points','FontSize',14,'interpreter','latex');
ylabel('$I$','FontUnits','points','FontSize',14,'interpreter','latex');
title(['$ p= $' num2str(p)...
'$\quad \lambda_0= $' num2str(L0)...
'$\quad  \lambda_1=$' num2str(L1) '$\quad  H(X_1+X_2)=$' num2str(-FF*log2(FF')),...
'$\quad  H( X_1,X_2)=$' num2str(-N*(q*log2(q)+p*log2(p))) ],'FontUnits','points','FontSize',14,'Interpreter','latex');
hold on
x = [t(1) t(end)];
y = [yy(1) dd1*t(end)];
line(x,y,'Color','blue','LineStyle','--')

hold on
x = [t(1) t(end)];
y = [I_u(1) dd2*t(end)];
line(x,y,'Color','red','LineStyle','--')
axis tight

h=legend('$I(X_1, X_2;Y_1, Y_2)$',...
'$I(X_1+X_2;Y_3)$','$\textrm{Tangent to blue curve at } T=0$','$\textrm{Tangent to red curve at } T=0 $');
set(h,'Location','southeast','FontUnits','points','FontSize',14,'Interpreter','latex','location','best');
grid on
%% fig
figure
```

```
plot(t,[N*MI;I_u],'LineWidth',2);
% hold on
% plot([t(1) t(end)],N*[-(q*log2(q)+p*log2(p)) ,-(q*log2(q)+p*log2(p))],'LineWidth',2)
xlabel('$T$','FontUnits','points','FontSize',14,'interpreter','latex');
% ylabel('$N \cdot I_1(X_1;Y_1)$','FontUnits','points','FontSize',14,'interpreter','latex');
ylabel('$I$','FontUnits','points','FontSize',14,'interpreter','latex');
title(['$ p= $' num2str(p)...
'$\quad \lambda_0= $' num2str(L0)...
'$\quad  \lambda_1=$' num2str(L1) '$\quad  T=$' num2str(t0)],'FontUnits','points','FontSize',14,'Interpreter','latex');
hold on
x = [t(1) t(end)];
y = [yy(1) dd1*t(end)];
line(x,y,'Color','blue','LineStyle','--')

hold on
x = [t(1) t(end)];
y = [I_u(1) dd2*t(end)];
line(x,y,'Color','red','LineStyle','--')
axis tight
hold on
plot([0 T],[-N*(q*log2(q)+p*log2(p))    -N*(q*log2(q)+p*log2(p)) ],'-m')
hold on
plot([0 T],[-FF*log2(FF')   -FF*log2(FF')],'-k')
h=legend('$I(X_1, X_2;Y_1, Y_2)$',...
'$I(X_1+X_2;Y_3)$','$\textrm{Tangent to blue curve at } T=0$','$\textrm{Tangent to red curve at } T=0 $',' $H(X_1,X_2)$',' $H(X_1+X_2)$');
```

```
set(h,'Location','southeast','FontUnits','points','↩
   FontSize',14,'Interpreter','latex','location','best');
grid on
axis([0 T 0 -N*(q*log2(q)+p*log2(p))  ])

toc
```

F.6 Code for Fig. (2.7) and Fig. (2.8). (Include code F.18.2)

```
%% Two bin Simulation
clear all
% close all
tic
%n=50;k=3;nchoosek(n+k-1,k-1)
T=1; % Total time available
v=[ ]; %initilize an empty matrix
p=0.125/2;
L0= 10;
L1=20;
global X
X=1;
d=0.02; % mathematica d = 0.1; T = 7; Sum[(d*x + d*y + d↩
   *z)/T*...
%   Boole[d*x + d*y + d*z == T], {x, 0, T/d}, {y, 0, T/d↩
   }, {z, 0, T/d}]
%  Matlab n=T/d;k=3;nchoosek(n+k-1,k-1)
%
tic
```

```
for i=0:d:T;
for j=0:d:T;
for k=0:d:T;

u=[i j k];

ff=sum(u,2);
if (abs(ff-T)< eps(16));  v=cat(1,v,u); end
end
end
end
toc
%  keyboard
a=zeros(1,length(v)); b=zeros(1,length(v));c=zeros(1,↩
   length(v));

% for i=1:length(v);
% [a(i)]=MI_2(p,L0,L1,v(i,1),v(i,2),v(i,3));  % Evaluate↩
    mutual information for each point (T1,T2,T3).
% % [a(i)]=MI_new(p,L0,L1,v(i,1),v(i,2),v(i,3));  % ↩
   Evaluate mutual information for each point (T1,T2,T3).
% % [a(i)]=my_MI_3(p,L0,L1,v(i,1),v(i,2),v(i,3));  % ↩
   Evaluate mutual information for each point (T1,T2,T3).
% end

a=arrayfun(@MI_2,p*ones(size(v,1),1),L0*ones(size(v,1)↩
   ,1),L1*ones(size(v,1),1),v(:,1),v(:,2),v(:,3));

toc
[e,f]=max(a)
v(f,:)

figure
set(gca,'FontSize',14)
```

```
scatter3(v(:,1),v(:,2),v(:,3),[ ],a,'filled');
hold on
scatter3(v(f,1),v(f,2),v(f,3),[ ],1,'+w');
colormap(jet);
axis equal;grid on;

h=colorbar; view(135,atand(1/sqrt(2)))
ylabel(h,'$I$','FontUnits','points','FontSize',14,'↩
   interpreter','latex');
xlabel('$T_1$','FontUnits','points','FontSize',14,'↩
   interpreter','latex');
ylabel('$T_2$','FontUnits','points','FontSize',14,'↩
   interpreter','latex');
zlabel('$T_3$','FontUnits','points','FontSize',14,'↩
   interpreter','latex');

title(['$ \scriptstyle{\mathop {\arg \max }\limits_{(T_1↩
   ,T_2,T_3)} I}=$(' num2str(v(f,1)) '$,$'   num2str(v(f↩
   ,2))...
'$,$'   num2str(v(f,3)) '$) \quad p= $' num2str(num2str(↩
   p,'%.6f')) ...
'$\quad \lambda_0= $' num2str(L0)...
'$\quad  \lambda_1=$' num2str(L1)],'FontUnits','points',↩
   'FontSize',14,'Interpreter','latex');
```

F.7 Code for Fig. (2.9) and Fig. (2.10). (Include code F.18.2)

```
% Probability of Detection(analytical), Correct Decision↩
    Rate(Empirical)
% and Mutual Information versus time T3, for Full Target↩
    Support

clear all
% close all

tic
p=0.125;     % Probability of 1 (higher rate)
%p=abs(-p+1);
T=1;      % Total time
 L0=0;    L1=1;     % Poisson rates
% L0=12.631578947368421;    L1=20;     % Poisson rates    ↩
   T3_0:  0.3535 ,  MI_0: 0.2504
%  L0=13.684210526315789;    L1=20;     % Poisson rates  ↩
   T3_0:  0.38384    ,  MI_0:  0.1828
q=1-p;

s=10^5;  % total number of samples / Data point.

global X
X=1;

D=100;  % Total number of data points.
v=[linspace(0,T/2,D)',linspace(0,T/2,D)',linspace(T,0,D)↩
   '];  % samples of (T1,T2,T3)

MI=[ ];
Cd=[ ];
Pd=[ ];


for i=1:length(v);
```

```
% [m,h0,h1,h2,h3]=MI_2(p,L0,L1,v(i,1),v(i,2),v(i,3));  %↩
    Evaluate mutual information for each point (T1,T2,T3)↩
   .
[m,h0,h1,h2,h3]=MI_new(p,L0,L1,v(i,1),v(i,2),v(i,3));  %↩
    Evaluate mutual information for each point (T1,T2,T3)↩
   .
% MI_new handles the L0=0 situation
MI=cat(2,MI,m);

%%
T1=v(i,1);  T2=v(i,2);  T3=v(i,3);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% e=eps(1);
% y1_max= poissinv(1-e,L1*T1);
% y2_max= poissinv(1-e,L1*T2);
% y3_max= poissinv(1-e,(L1+L1)*T3);
%
% [y1,y2,y3]=meshgrid(0:y1_max,0:y2_max,0:y3_max);
%
% % Conditional Probabilities of Y1 given X
% py1_00=poisspdf(y1,L0*T1);
% py1_01=py1_00;
% py1_10=poisspdf(y1,L1*T1);
% py1_11=py1_10;
%
% % Conditional Probabilities of Y2 given X
% py2_00=poisspdf(y2,L0*T2);
% py2_10=py2_00;
% py2_01=poisspdf(y2,L1*T2);
% py2_11=py2_01;
%
% % Conditional Probabilities of Y3 given X
% py3_00=poisspdf(y3,(L0+L0)*T3);
% py3_01=poisspdf(y3,(L0+L1)*T3);
```

```
% % py3_10=poisspdf(y3,(L1+L0)*T3);
% py3_10=py3_01;
% py3_11=poisspdf(y3,(L1+L1)*T3);
%
% % Four hypotheses
%  H00=q^2*(py1_00.*py2_00.*py3_00);
%  H01=q*p*(py1_01.*py2_01.*py3_01);
%  H10=p*q* (py1_10.*py2_10.*py3_10);
% % Note: H01 is NOT equal to H10.
%  H11=p^2*(py1_11.*py2_11.*py3_11);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Method 1 to compute MAP analytical Pd (old method of ↩
   2015. It is correct but slow)
% H00=h0; H01=h1; H10=h2; H11=h3;
%
%  % Maximum values in H00 H01 H10 H11
%  m1=max(H00,H01);  m2=max(m1,H10);
%  mx=max(m2,H11);  % Maximum value in H00 H01 H10 H11
%
%  M1=(H00==mx);  M2=(H01==mx);
%  M3=(H10==mx);  M4=(H11==mx);
%
%  % Any 4 hypothesis be equal
% J1=and(M1,and(M2,and(M3,M4)));
% K1=sum(sum(sum(H00.*J1)));
% M1=(~J1).*(M1);
% M2=(~J1).*(M2);
% M3=(~J1).*(M3);
% M4=(~J1).*(M4);
%
% % Any 3 hypothesis be equal
% J2=and(M1,and(M2,M3));
% K2=sum(sum(sum(H00.*J2)));
```

```
% J3=and(M1,and(M2,M4));
% K3=sum(sum(sum(H00.*J3)));
% J4=and(M1,and(M3,M4));
% K4=sum(sum(sum(H00.*J4)));
% J5=and(M2,and(M3,M4));
% K5=sum(sum(sum(H01.*J5)));
% M1=(~J2).*(~J3).*(~J4).*(M1);
% M2=(~J2).*(~J3).*(~J5).*(M2);
% M3=(~J2).*(~J4).*(~J5).*(M3);
% M4=(~J3).*(~J4).*(~J5).*(M4);
%
% % Any 2 hypothesis be equal
% J6=and(M1,M2);
% K6=sum(sum(sum(H00.*J6)));
% J7=and(M1,M3);
% K7=sum(sum(sum(H00.*J7)));
% J8=and(M1,M4);
% K8=sum(sum(sum(H00.*J8)));
% J9=and(M2,M3);
% K9=sum(sum(sum(H01.*J9)));
% J10=and(M2,M4);
% K10=sum(sum(sum(H01.*J10)));
% J11=and(M3,M4);
% K11=sum(sum(sum(H10.*J11)));
% M1=(~J6).*(~J7).*(~J8).*(M1);
% M2=(~J6).*(~J9).*(~J10).*(M2);
% M3=(~J7).*(~J9).*(~J11).*(M3);
% M4=(~J8).*(~J10).*(~J11).*(M4);
%
% % None been equal
% K12=sum(sum(sum(H00.*M1)));
% K13=sum(sum(sum(H01.*M2)));
% K14=sum(sum(sum(H10.*M3)));
% K15=sum(sum(sum(H11.*M4)));
```

```
%
% d=K1+K2+K3+K4+K5+K6+K7+K8+K9+K10+...
%     K11+K12+K13+K14+K15; %Probability of correct ↩
   Decisions
%
% Pd=cat(2,Pd,d);

%% Another method to compute MAP's Pd  (New Method of 1↩
   st Nov 2020)
F00=((h0 >= h1) & (h0 >= h2) & (h0>= h3));
F01=((h1 > h0) & (h1 >= h2) &  (h1>= h3));
F10=((h2 >h0) & (h2 > h1) &  (h2>= h3));
F11=((h3 > h0) & (h3 > h1) &  (h3> h2));

d3=sum(sum(sum(h0.*F00)))+sum(sum(sum(h1.*F01)))+sum(sum↩
   (sum(h2.*F10)))+sum(sum(sum(h3.*F11)));
Pd=cat(2,Pd,d3);


%% Empirical Correct Decision Rate
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% s is total samples /total ensembles
b=binornd(1,p,s,2);  % sx2 Bernoulli trials
B=b;
%
b=L1*b;  % replace 1 with L1 (logical indexing)
b(b(:,:)==0)=L0; % replace 0 with L0

% U=[poissrnd(L0*T1,s,1) poissrnd(L1*T2,s,1)  poissrnd((↩
   L0+L1)*T3 ,s,1)];

x=poissrnd(T1*b(:,1),s,1);    % s no. of samples from ↩
   poisson distributed rvs
y=poissrnd(T2*b(:,2),s,1);
```

```
z=poissrnd(T3*(b(:,1)+b(:,2)),s,1);


% Four hypotheses
H = [q^2*poisspdf(x,L0*T1).*poisspdf(y,L0*T2).*poisspdf(z,(L0+L0)*T3),...
    q*p*poisspdf(x,L0*T1).*poisspdf(y,L1*T2).*poisspdf(z,(L0+L1)*T3), ...
    p*q*poisspdf(x,L1*T1).*poisspdf(y,L0*T2).*poisspdf(z,(L1+L0)*T3) ,...
    p^2*poisspdf(x,L1*T1).*poisspdf(y,L1*T2).*poisspdf(z,(L1+L1)*T3) ];

[mm,k]=max(H,[ ],2);  %Column vector of maximum values along row
M=[H(:,1)==mm H(:,2)==mm H(:,3)==mm H(:,4)==mm]; %Location of max. values in H


%% Method 1. Empirical Pd. Flipping a coin to select one outcome (Old method of  2015)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% N=sum(M,2);
% for k=1:length(M);
% if N(k)==2;    f=find(M(k,:));   u=unidrnd(2);
%     if u==1;    M(k,:)=0;    M(k,f(1))=1;
%     elseif  u==2;    M(k,:)=0;     M(k,f(2))=1;
%     end
% elseif N(k)==3;    f=find(M(k,:));      u=unidrnd(3);
%     if u==1;     M(k,:)=0;       M(k,f(1))=1;
%     elseif  u==2;     M(k,:)=0;       M(k,f(2))=1;
%     elseif  u==3;      M(k,:)=0;       M(k,f(3))=1;
%     end
% elseif N(k)==4;    f=find(M(k,:));    u=unidrnd(4);
```

```
%      if u==1; M(k,:)=0; M(k,f(1))=1;
%      elseif  u==2; M(k,:)=0; M(k,f(2))=1;
%      elseif  u==3; M(k,:)=0; M(k,f(3))=1;
%      elseif  u==4; M(k,:)=0; M(k,f(4))=1;
%      end
% end
% end
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  Tar=zeros(4,s);
%  for j=1:s
%      if  isequal(b(j,:),[L0 L0]);
%      Tar(1,j)=1;
%      elseif   isequal(b(j,:),[L0 L1]);
%      Tar(2,j)=1;
%      elseif   isequal(b(j,:),[L1 L0]);
%      Tar(3,j)=1;
%      elseif  isequal(b(j,:),[L1 L1]);
%      Tar(4,j)=1;
%      end
%  end
%
% % plotconfusion(Tar,H')
% % plotroc(Tar,M')
% %u=unidrnd(2,1,1)-1
% %  [C,CM,IND,PER] = confusion(Tar,M');
% %  Pm=cat(2,Pm,C);
%
% Tp=zeros(1,s);
% Z=M';
% for r=1:s;
% Tp(r)=isequal(Tar(:,r),Z(:,r));  %Total correct ↩
   answers
% end
```

```
% su=sum(Tp)/s;
% Cd=cat(2,Cd,su);

%% Method 2. Empirical Pd. Flipping a coin to select one outcome (New Method of 1st Nov 2020)

% Just pick the first 1 along every row in Matrix M. That's it !
D0=M.*(cumsum(M,2)<2); % One '1' per row
%
% D1=M.*(cumsum(M,2)<2 & (sum(M,2)==1)); % One '1' per row
% D2=M.*(cumsum(M,2)<2 & (sum(M,2)==2)); % Two '1' per row
% D3=M.*(cumsum(M,2)<2 & (sum(M,2)==3)); % Three '1' per row
% D4=M.*(cumsum(M,2)<2 & (sum(M,2)==4)); % Four '1' per row

T1=zeros(s,1);
T1(b(:,1)==L0 & b(:,2)==L0)=1;
T2=zeros(s,1);
T2(b(:,1)==L0 & b(:,2)==L1)=1;
T3=zeros(s,1);
T3(b(:,1)==L1 & b(:,2)==L0)=1;
T4=zeros(s,1);
T4(b(:,1)==L1 & b(:,2)==L1)=1;
Targets=[T1 T2 T3 T4];

% plotconfusion(Tar,H')
% plotroc(Tar,M')
%u=unidrnd(2,1,1)-1
%  [C,CM,IND,PER] = confusion(Tar,M');
Tp=Targets.*D0;  %Total correct answers
```

```
su=sum(sum(Tp))/s;
Cd=cat(2,Cd,su);
end

%%
[f1,g1]=max(MI);  % maximum value of MI
v(g1,:)

[f2,g2]=max(Pd);   % maximum value of Pd
v(g2,:)

[f3,g3]=max(Cd);
v(g3,:)

ft=14;  % Font size of labels + legends + axis
% Individual Figures plots
%%
figure;
plot(v(:,3),MI, 'LineWidth',2);
set(gca,'FontSize',ft)
 xlabel('$T_3$','FontUnits','points','FontSize',ft,'↩
    Interpreter','latex');
 ylabel('$I(X_1,X_2;Y_1,Y_2,Y_3)$','FontUnits','points',↩
    'FontSize',ft,'Interpreter','latex');
%  title(['$\scriptstyle{\mathop {\arg \max }\limits_{\↩
   gamma} }\displaystyle I = $'...
%      num2str(v(g1,3))  '$\quad p= $' num2str(p)...
%      '$\quad \lambda_0= $' num2str(L0)...
%      '$\quad  \lambda_1=$' num2str(L1) ...
%      '$\quad T=$' num2str(T) '$\quad \scriptstyle{\↩
   mathop {\arg \max }\limits_{\gamma} }\displaystyle P_d↩
    = $'...
%      num2str(v(g2,3)) ],'FontUnits','points','FontSize↩
   ',ft,'Interpreter','latex');
```

```
 title(['$\mathop {\arg \max }\limits_{T_3} I = $'...
     num2str(v(g1,3))  '$\quad p= $' num2str(p)...
     '$\quad \lambda_0= $' num2str(L0)...
     '$\quad  \lambda_1=$' num2str(L1) ...
     '$\quad T=$' num2str(T) '$\quad \mathop {\arg \max ↩
       }\limits_{\gamma}  P_d = $'...
     num2str(v(g2,3)) ],'FontUnits','points','FontSize',↩
       ft,'Interpreter','latex');
 grid on
%  axis tight

%%
 figure;
 plot(v(:,3),Pd,'b', 'LineWidth',2);
 set(gca,'FontSize',ft)
xlabel('$T_3$','FontUnits','points','FontSize',ft,'↩
   Interpreter','latex');
 ylabel('$P_d$','FontUnits','points','FontSize',ft,'↩
    Interpreter','latex');
 title(['$p= $' num2str(p)...
     '$\quad \lambda_0= $' num2str(L0)...
     '$\quad  \lambda_1=$' num2str(L1) ...
     '$\quad T=$' num2str(T)],'FontUnits','points','↩
       FontSize',ft,'Interpreter','latex');
%   axis tight

  hold on; plot(v(:,3),Cd, 'r', 'LineWidth',2);
 xlabel('$T_3$','FontUnits','points','FontSize',ft,'↩
    Interpreter','latex');
 ylabel ('$C_d\: $, $\: P_d$','FontUnits','points','↩
    FontSize',ft,'Interpreter','latex');

 title(['$p= $' num2str(p)...
     '$\quad \lambda_0= $' num2str(L0)...
```

```
      '$\quad  \lambda_1=$' num2str(L1) ...
      '$\quad T=$' num2str(T) '$\quad \mathop {\arg \max }\limits_{T_3} P_d = $' ...
      num2str(v(g2,3)) '$\quad \rm{samples}= $' num2str(s)],'FontUnits','points','FontSize',ft,'Interpreter','latex');

 h=legend('Analytical ($Pd$)','Empirical ($Cd$)');
set(h,'FontUnits','points','FontSize',ft, 'Location','SouthWest','Interpreter','latex');
axis([0 T 0 1]);
grid on

 %%
 % Double axis plot for MI, Pd and Cd.
 figure
 E=[q^2  p*q p*q p^2]; Hh=-E*log2(E');
%  [AX,H1,H2]=plotyy([v(:,3)',v(:,3)'],[MI,H*ones(1,length(v(:,3)))],v(:,3),Pd);
 [AX,H1,H2]=plotyy(v(:,3)',[MI;Hh*ones(1,length(v(:,3)))],v(:,3),Pd);
 set(gca,'FontSize',ft);
%   axis tight

axis(AX(1),'tight');
axis(AX(2),'tight');


%   title(['$\mathop {\arg \max }\limits_{\gamma} }\displaystyle I=$'  num2str(v(g1,3))...
%     '$ \quad p= $'  num2str(p)    '$ \quad \lambda_0= $' num2str(L0)...
%      '$\quad  \lambda_1=$' num2str(L1) ...
```

```
%       '$\quad T=$' num2str(T)  '$\quad \scriptstyle{\↩
   mathop {\arg \max }\limits_{\gamma}} \displaystyle P_d↩
    = $'...
%      num2str(v(g2,3))],'FontUnits','points','FontSize↩
   ',ft,'Interpreter','latex');
  title(['$\mathop {\arg \max }\limits_{T_3}  I=$'  ↩
     num2str(v(g1,3))...
    '$ \quad p= $'  num2str(p)    '$ \quad \lambda_0= $'↩
        num2str(L0)...
     '$\quad  \lambda_1=$' num2str(L1) ...
     '$\quad T=$' num2str(T)  '$\quad \mathop {\arg \max↩
         }\limits_{T_3} P_d = $'...
     num2str(v(g2,3))],'FontUnits','points','FontSize',↩
        ft,'Interpreter','latex');

xlabel('$T_3$','FontUnits','points','FontSize',ft,'↩
   Interpreter','latex');
ylabel(AX(1),'$I \: ,\: H$','FontUnits','points','↩
   FontSize',ft,'Interpreter','latex'); % left y-axis
ylabel(AX(2),'$P_d$','FontUnits','points','FontSize',ft,↩
   'Interpreter','latex'); % right y-axis


set(AX(1),'Position', [0.13 0.11 0.775-.06 0.815]);
set(AX(2),'Position', [0.13 0.11 0.775-.06 0.815]);
% Original position was [0.13 0.11 0.775 0.815]
% Applied change in width: "-.08". Choose as desired

set( H1,'LineWidth',2);
set(H1(1,1),'LineStyle','-','Color','blue');
set(H1(2,1),'LineStyle','--','Color','blue');

set( H2,'LineWidth',2);
set(H2,'LineStyle','-.','Color','red');
```

```
set(AX,{'ycolor'},{'b';'r'});
 h=legend('$I(X_1,X_2;Y_1,Y_2,Y_3)$',['$H(X_1,X_2)=$' ↩
    num2str(Hh)],'$P_d$');
 set(h,'FontUnits','points','FontSize',ft, 'Location','↩
    SouthWest','Interpreter','latex');

 % Set the number of ticks on two Y axes
 NumTicks = 5;
L = get(AX(1),'YLim');
set(AX(1),'YTick',linspace(L(1),L(2),NumTicks),'FontSize↩
   ',ft);

 NumTicks = 5;
L = get(AX(2),'YLim');
set(AX(2),'YTick',linspace(L(1),L(2),NumTicks),'FontSize↩
   ',ft);
 grid on
% set(gcf, 'units','normalized','outerposition',[0 0 1 ↩
   1]);
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  toc
```

F.8 Code for Fig. (2.11) and Fig. (2.12)

```
% Mutual Information versus time a3, for Full Target ↩
   Supports
% works with arrayfunc ( ). To use it for GPU

% Elapsed time is 766.434650 seconds with arrayfun ( )
```

```
% Elapsed time is  807.016181 seconds. seconds seconds ↩
   with for loop
clear all
% close all

tic
% s=10^4;  % total number of samples / Data point.
% global X Umax y1 y2 y3 q

global X K D
X=1; % Total Iterations := (K^2-K)/2 * D

%% Parameters

% P=1/16;
% P=1/8;
 P=1/4;
% P=1/2;
% P=1-1/16;
%  P=1-1/1024;


% q=1-P;
% T=10;     % Total time
D=200;  % Total number of data points (T1,T2,T3) for ↩
   each search for optimal value.
Tmax=1; % Maximum total time
v=[linspace(0,Tmax/2,D)',linspace(0,Tmax/2,D)',linspace(↩
   Tmax,0,D)'];  % samples of (T1,T2,T3)
%% Grid formation
 Umax=5; % Max Limit of L1*T
 Umin=0; % Min Limit of L1*T

K=20; % K x K grid
```

```
C1=linspace(Umin,Umax,K);
[A1,B1]=meshgrid(C1,C1); %X1 := L1.T,    X2:=L0.T

X1=A1;
Y1=B1;

X1(A1<=B1)=[ ]; % Eliminate the region (L1T<=L0T). When element is eliminated Matlab results the row vector instead of matrix
Y1(A1<=B1)=[ ]; % Eliminate the region (L1T<=L0T).

% X1(A1<=B1)=0; % Replace every element in matrix X1 with 0 whenever X1<=Y1
% Y1(A1<=B1)=0; % Replace every element in matrix Y1 with 0 whenever X1<=Y1

% %% my_MI_2.m grid (This grid is fixed for all values of L0 and L1). So calculated only once and for all.
% e=eps(0.5);
% y1_max=poissinv(1-e,Umax);
% y3_max=poissinv(1-e,2*Umax);
% [y1,y2,y3]=meshgrid(0:y1_max,0:y1_max,0:y3_max);

%% Computations start
T3_0=zeros(size(X1,1),size(X1,2)); % Zero Matrix containing the optimal T3 value for each pair of (L1T,L0T)
MI_0=zeros(size(X1,1),size(X1,2)); % Zero Matrix containing the optimal T3 value for each pair of (L1T,L0T)
for k=1:size(X1,1)
for l=1:size(X1,2)
% if X1(k,l)==0 % To avoid running the optimization in the region L1T<=L0T
```

```
% T3_O(k,l)=0;
% else
% Tmax=X1(k,l);
% v=[linspace(0,Tmax/2,D)',linspace(0,Tmax/2,D)',↩
   linspace(Tmax,0,D)'];  % samples of (T1,T2,T3)
% sum(v,2)
% keyboard

% if X==2
%  keyboard
% end
N=length(P); %Number of probability points
O_MI=[ ];
O_T3=[ ];
% O_p= [ ];
% m_info=[ ];
for j=1:N
p=P(j);    % Probability of 1 (higher rate)
% q=1-p;
% MI=[ ];

% for i=1:length(v);
% [m,h0,h1,h2,h3]=MI_new(p,Y1(k,l),X1(k,l),v(i,1),v(i,2)↩
   ,v(i,3));  % Evaluate mutual information for each ↩
   point (T1,T2,T3).
%  [m,h0,h1,h2,h3]=MI_2(p,Y1(k,l),X1(k,l),v(i,1),v(i,2),↩
   v(i,3));  % Evaluate mutual information for each point↩
    (T1,T2,T3).
% [m,h0,h1,h2,h3]=MI_2(p,Y1(k,l)/X1(k,l),1,v(i,1),v(i,2)↩
   ,v(i,3));  % Evaluate mutual information for each ↩
   point (T1,T2,T3).
% [m,h0,h1,h2,h3]=new_MI_2(p,Y1(k,l)/X1(k,l),1,v(i,1),v(↩
   i,2),v(i,3));  % Evaluate mutual information for each ↩
   point (T1,T2,T3).
```

```
% [m,h0,h1,h2,h3]=My_MI(p,Y1(k,l)/X1(k,l),1,v(i,1),v(i,2),v(i,3));  % Evaluate mutual information for each point (T1,T2,T3).
% [m]=my_MI_2(p,Y1(k,l)/X1(k,l),1,v(i,1),v(i,2),v(i,3));  % Evaluate mutual information for each point (T1,T2,T3).
% [m]=my_MI_2(p,Y1(k,l),X1(k,l),v(i,1),v(i,2),v(i,3));  % Evaluate mutual information for each point (T1,T2,T3).
% m=arrayfun(@my_MI_3,gpuArray(p*ones(size(v,1),1)),gpuArray(Y1(k,l)*ones(size(v,1),1)),gpuArray(X1(k,l)*ones(size(v,1),1)),gpuArray(v(:,1)),gpuArray(v(:,2)),gpuArray(v(:,3)));
% It is under construction
m=arrayfun(@my_MI_3,p*ones(size(v,1),1),Y1(k,l)*ones(size(v,1),1),X1(k,l)*ones(size(v,1),1),v(:,1),v(:,2),v(:,3)); % It works great. Faster than for loop.
% [m]=my_MI_3(p,Y1(k,l),X1(k,l),v(i,1),v(i,2),v(i,3));  % Evaluate mutual information for each point (T1,T2,T3).

% MI=cat(2,MI,m);
 MI=m;
% end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[f1,g1]=max(MI);  % maximum value of MI
% v(g1,:)
% m_info=cat(1,m_info,MI); % size: length(P) X D
O_MI=cat(2,O_MI,f1);   % Optimal MI  size: 1 X length(P)
O_T3=cat(2,O_T3,v(g1,3));    % Optimal T3 size: 1 X length(P)
end
T3_O(k,l)=O_T3;
MI_O(k,l)=O_MI;
```

```
% end
end
end
H=-[1-p p]*log2([1-p p]')*2; % H ( X )
 %%
% figure
% scatter(X1,Y1,[ ] ,'filled')
% ylabel('$\lambda_0 \cdot T$','Interpreter','latex');
% xlabel('$\lambda_1 \cdot T$','Interpreter','latex');
% title({['$ \rm{Grid \: for \:} \lambda_1 T> \lambda_0 ↩
   T $']},'Interpreter','latex');
% axis tight
% grid on
%%
figure
% nexttile
scatter(X1,Y1,100,MI_0 ,'filled');
ylabel('$\lambda_0 \cdot T$','FontSize',14,'Interpreter'↩
   ,'latex');
xlabel('$\lambda_1 \cdot T$','FontSize',14,'Interpreter'↩
   ,'latex');
% title({['$ p=$'...
%     num2str(p) '$\quad T=$' num2str(Tmax) '$\quad H(X)↩
   =$' num2str(H)],...
%     ['$ (T_1,T_2,T_3) := (\frac{T-\alpha}{2},\frac{T-\↩
   alpha}{2},\alpha); \quad 0 \le \alpha \le T.$']},'↩
   Interpreter','latex');

title({['$ p=$'...
    num2str(p) '$\quad H(X)=$' num2str(H)]},'FontSize'↩
       ,14,'Interpreter','latex');

caxis([0  max(MI_0)]); % maps blue to 0 and red to ↩
   maximum value
```

```
colormap(jet(256));
h=colorbar;
% h.Limits = [0  max(MI_0)];
% set(h, 'ylim', [min(MI_0) max(MI_0)])
ylabel(h,'$I^o(X;Y)$','FontUnits','points','FontSize'↩
   ,14,'interpreter','latex');
axis tight
grid on
%%
figure
% nexttile
% scatter(reshape(X1,1,numel(X1)),reshape(Y1,1,numel(Y1)↩
   ),50,reshape(T3_0,1,numel(T3_0)),'filled')
% X1(T3_0==0)=[ ];
% Y1(T3_0==0)=[ ];
% T3_0(T3_0==0)=[ ];

scatter(X1,Y1,100,T3_0,'s','filled');
ylabel('$\lambda_0 \cdot T$','FontSize',14,'Interpreter'↩
   ,'latex');
xlabel('$\lambda_1 \cdot T$','FontSize',14,'Interpreter'↩
   ,'latex');
% title({['$ \rm{Optimal \: joint \: sensing \: time \: ↩
   (\alpha^o) \: in \: region \:} \lambda_1 T> \lambda_0 ↩
   T $'],...
%     ['$p=$' num2str(p) '$\quad H(X)=$' num2str(H)]},'↩
   Interpreter','latex');

title({['$p=$' num2str(p) '$\quad H(X)=$' num2str(H)]},'↩
   FontSize',14,'Interpreter','latex');

caxis([0  Tmax]); % maps blue to 0 and red to maximum ↩
   value
h=colorbar;
```

```
% h.Limits = [0  Tmax];
colormap(jet(256));
% h=colorbar;
% h.Limits = [0  Tmax];
% set(h, 'ylim', [min(T3_O) max(T3_O)])
ylabel(h,'$\alpha^o$','FontUnits','points','FontSize'↩
   ,14,'interpreter','latex');
axis tight;
grid on;
%%
BW=T3_O;
BW(T3_O >0 )=1;
BW(T3_O == 0)=0;
figure
% scatter(X1,Y1,100,BW,'s','filled')
gscatter([X1 max(X1) max(X1)],[Y1 -1 -2 ],[BW  0 1],'rb'↩
   ,[ ],35) % Point (max(X1),-1) is included to make the ↩
   extra group visible in legend
ylabel('$\lambda_0 \cdot T$','FontSize',14,'Interpreter'↩
   ,'latex');
xlabel('$\lambda_1 \cdot T$','FontSize',14,'Interpreter'↩
   ,'latex');
title({['$p=$' num2str(p) '$\quad H(X)=$' num2str(H)]},'↩
   FontSize',14,'Interpreter','latex');
% caxis([0  1]); % maps blue to 0 and red to maximum ↩
   value
% h=colorbar;
% colormap(gray(2));
% ylabel(h,'$\alpha^o$','FontUnits','points','FontSize↩
   ',14,'interpreter','latex');
h=legend('$\alpha^o = 0 $','$\alpha^o > 0$');
set(h,'FontUnits','points','FontSize',14,'Interpreter','↩
   latex');
axis tight;
```

```
axis([min(X1) max(X1) min(Y1) max(Y1)]);
grid on;
%%
cmap = colormap;
s1=length(unique(cmap,'rows')); % no. of distinct colors used in cmap
formatSpec1 = 'no. of distinct colors used in cmap:  \t\t%d  \n';
fprintf(formatSpec1,s1)
s2=length(unique(T3_O)); % no. of unique optimal T3 values
formatSpec2 = 'no. of distinct values in T3_O : \t\t%d  \n';
fprintf(formatSpec2,s2)
s3=nnz(T3_O); % no. of non zero elements in optimal T3
formatSpec3 = 'no. of non-zero values in T3_O :       \t\t%d  \n';
fprintf(formatSpec3,s3)
s4= sum(T3_O==Tmax); % no. of ones in optimal T3
formatSpec4 = 'no. of ones in T3_O : \t\t%d  \n';
fprintf(formatSpec4,s4)
s5= sum(T3_O==0); % no. of zeros in optimal T3
formatSpec5 = 'no. of zeros in T3_O : \t\t%d  \n';
fprintf(formatSpec5,s5)
s6=numel(T3_O); % no. of zeros in optimal T3
formatSpec6 = 'no. of elements in T3_O : \t\t%d  \n';
fprintf(formatSpec6,s6)

toc
```

F.9 Code for Fig. (3.2). (Include code F.18.4)

```
% Gaussian Monte Carlo MI vs. T3 and Pd vs. T3 for Full ↩
   Target Support
clear all
% close all
tic
global X N C NN iter % It is introduced to count the ↩
   number of function calls.
X=1; % Total Iterations := (K^2-K)/2 * D
N=2; % Number of bins/targets
C=diag([1 1 1]);  %Covariance matrix of component ↩
   multivariate Gaussian
NN=10^7; % Monte Carlo Samples per dimension

%% Parameters
% P=1/16;
% P=1/8;
 p=0.125/8;
 L0=1;   L1=20;
 T=1; % Maximum total time
% P=1/2;
% P=1-1/16;
%  P=1-1/1024;
% q=1-P;
% T=10;     % Total time
D=400;  % Total number of data points (T1,T2,T3) for ↩
   each search for optimal value.
v=[linspace(0,T/2,D)',linspace(0,T/2,D)',linspace(T,0,D)↩
   '];  % samples of (T1,T2,T3) % Constrained
```

```
% v=[linspace(0,T,D)',linspace(0,0,D)',linspace(0,0,D)']; % samples of (T1,T2,T3) % UnConstrained
 K=1; % K x K grid
 q=1-p;
iter=D;

MI=[ ];
Cd=[ ];
% Pd=[ ];


% for i=1:length(v);
% [m,h0,h1,h2,h3]=MI_2(p,L0,L1,v(i,1),v(i,2),v(i,3));  % Evaluate mutual information for each point (T1,T2,T3).
% [m,h0,h1,h2,h3]=MI_new(p,L0,L1,v(i,1),v(i,2),v(i,3)); % Evaluate mutual information for each point (T1,T2,T3).
[m,idn]=arrayfun(@Gauss_MI_2,p*ones(size(v,1),1),L0*ones(size(v,1),1),L1*ones(size(v,1),1),sqrt(v(:,1)),sqrt(v(:,2)),sqrt(v(:,3))); % It works great. Faster than for loop.

% MI_new handles the L0=0 situation
MI=cat(2,MI,m);
Cd=cat(2,Cd,idn/NN);

%%
[f1,g1]=max(MI);  % maximum value of MI
v(g1,:)

% [f2,g2]=max(Pd);   % maximum value of Pd
% v(g2,:)
```

```
[f3,g3]=max(Cd);
v(g3,:)

ft=14;  % Font size of labels + legends + axis

%%
% figure;
% plot(v(:,3),MI, 'LineWidth',2);
% set(gca,'FontSize',ft)
%  xlabel('$T_3$','FontUnits','points','FontSize',ft,'↩
   Interpreter','latex');
%  ylabel('$I(X_1,X_2;Y_1,Y_2,Y_3)$','FontUnits','points↩
   ','FontSize',ft,'Interpreter','latex');
%  title(['$\mathop {\arg \max }\limits_{T_3} I = $'...
%      num2str(v(g1,3))  '$\quad p= $' num2str(p)...
%      '$\quad \lambda_0= $' num2str(L0)...
%      '$\quad  \lambda_1=$' num2str(L1) ...
%      '$\quad T=$' num2str(T) '$\quad \mathop {\arg \↩
   max }\limits_{T_3}  C_d = $'...
%      num2str(v(g3,3)) ],'FontUnits','points','FontSize↩
   ',ft,'Interpreter','latex');
%  grid on

 %%
 figure
 E=[q^2  p*q p*q p^2]; Hh=-E*log2(E');
yyaxis left
% ax(1)=plot(v(:,3)',[  MI'  ;  Hh*ones(1,length(v(:,3))↩
   )], 'LineWidth',2);
plot(v(:,3)',[  MI'  ;  Hh*ones(1,length(v(:,3)))], '↩
   LineWidth',2);
ylabel('$I \: ,\: H$','FontUnits','points','FontSize',ft↩
   ,'Interpreter','latex'); % left y-axis
ylim([0 Hh])
```

```
yyaxis right
% ax(2)=plot(v(:,3)',Cd', 'LineWidth',2);
plot(v(:,3)',Cd', 'LineWidth',2);
ylabel('$P_d$','FontUnits','points','FontSize',ft,'↩
   Interpreter','latex'); % right y-axis
ylim([0 1])
set(gca,'FontSize',ft);
% linkaxes(ax,'x');
  title(['$\mathop {\arg \max }\limits_{T_3}  I=$'  ↩
     num2str(v(g1,3))...
    '$ \quad p= $'  num2str(p)    '$ \quad \lambda_0= $'↩
       num2str(L0)...
     '$\quad  \lambda_1=$' num2str(L1) ...
     '$\quad T=$' num2str(T)  '$\quad \mathop {\arg \max↩
        }\limits_{T_3} P_d = $'...
     num2str(v(g3,3))],'FontUnits','points','FontSize',↩
       ft,'Interpreter','latex');

xlabel('$T_3$','FontUnits','points','FontSize',ft,'↩
   Interpreter','latex');
 h=legend('$I(X_1,X_2;Y_1,Y_2,Y_3)$',['$H(X_1,X_2)=$' ↩
    num2str(Hh)],'$P_d$');
 set(h,'FontUnits','points','FontSize',ft, 'Location','↩
    SouthWest','Interpreter','latex');
 grid on
  toc
```

F.10 Code for Fig. (3.3). (Include code F.18.5)

```
%% Two bin Simulation
clear all
% close all
tic
global X N C NN iter
X=1;
%n=50;k=3;nchoosek(n+k-1,k-1)
T=10; % Total time available
N=2; % Number of bins/targets
C=diag([1 1 1]);  %Covariance matrix of component ↩
   multivariate Gaussian
NN=10^6; % Monte Carlo Samples per dimension
% v=[ ]; %initialize an empty matrix
% p=0.125/2;
p=0.99;
q=1-p;
L0=0;
L1=2;

%% (T1,T2,T3) Grid Generation

%  Method 1

% d=0.02; % mathematica d = 0.1; T = 7; Sum[(d*x + d*y +↩
    d*z)/T*...
% %   Boole[d*x + d*y + d*z == T], {x, 0, T/d}, {y, 0, T↩
   /d}, {z, 0, T/d}]
% %  Matlab n=T/d;k=3;nchoosek(n+k-1,k-1)
% %
% for i=0:d:T;
% for j=0:d:T;
% for k=0:d:T;
%
% u=[i j k];
```

```
%
% ff=sum(u,2);
% if (abs(ff-T)< eps(16));  v=cat(1,v,u); end
% end
% end
% end

% Method 2

%% Triangular grid generation from 2D-meshgrid.
d=0.05;
[x,y]=meshgrid(0:d:T,0:d:T);
u=(((x+y)<= T) & (y <= x)); % union of two regions: the first norm <= T and y<=x. This is the right anlge triangle.
x=x.*u;
y=y.*u;

x(x==0 & u==0)=NaN;  % to NOT let exclude the points (x,0). NaN is used to later remove it from array.
y(y==0 & u==0)=NaN; % to NOT let exclude the points (x,0)

w=(isnan(x)).*(isnan(y)); % One in our region of interest.

z=T-(x+y); % generate z-co-ordinates by T=T1+T2+T3

x=reshape(x,1,numel(x)); % To removes NaNs we first have reshaped
y=reshape(y,1,numel(y));
z=reshape(z,1,numel(z));

x=rmmissing(x); % remove NaNs
```

```
y=rmmissing(y);
z=rmmissing(z);

v= [x' y' z']; % Res
iter= length(v);
%% Computation of MI

% a=zeros(1,length(v)); b=zeros(1,length(v));c=zeros(1,↩
   length(v));

% for i=1:length(v);
% [a(i)]=MI_2(p,L0,L1,v(i,1),v(i,2),v(i,3));  % Evaluate↩
    mutual information for each point (T1,T2,T3).
% % [a(i)]=MI_new(p,L0,L1,v(i,1),v(i,2),v(i,3));  % ↩
   Evaluate mutual information for each point (T1,T2,T3).
% % [a(i)]=my_MI_3(p,L0,L1,v(i,1),v(i,2),v(i,3));  % ↩
   Evaluate mutual information for each point (T1,T2,T3).
% end
% a=arrayfun(@Gauss_MI_2,p*ones(size(v,1),1),L0*ones(↩
   size(v,1),1),L1*ones(size(v,1),1),v(:,1),v(:,2),v(:,3)↩
   );
a=arrayfun(@Gauss_MI,p*ones(size(v,1),1),L0*ones(size(v↩
   ,1),1),L1*ones(size(v,1),1),sqrt(v(:,1)),sqrt(v(:,2)),↩
   sqrt(v(:,3))); % sqrt(v)


m1=(x'==y'); % To find maximum only when T1=T2
[e,f]=max(m1.*a);
v(f,:)

% [e,f]=max(a);
% v(f,:)
```

```
toc
%% Plotting
figure
set(gca,'FontSize',14)
% scatter3(v(:,1),v(:,2),v(:,3),[ ],a,'filled');
scatter3([  v(:,1) ; v(:,2)  ]  ,  [   v(:,2)  ;  v(:,1)↩
      ]  ,  [ v(:,3)  ;  v(:,3)],   [ ]    ,   [a   ;   ↩
   a] ,'filled'); % Symmetry induced in the plot
hold on
scatter3(v(f,1),v(f,2),v(f,3),[ ],1,'+w');
colormap(jet(256));
axis equal;grid on;

h=colorbar; view(135,atand(1/sqrt(2))) ;
ylabel(h,'$I$','FontUnits','points','FontSize',14,'↩
   interpreter','latex');
xlabel('$T_1$','FontUnits','points','FontSize',14,'↩
   interpreter','latex');
ylabel('$T_2$','FontUnits','points','FontSize',14,'↩
   interpreter','latex');
zlabel('$T_3$','FontUnits','points','FontSize',14,'↩
   interpreter','latex');

title(['$ \scriptstyle{\mathop {\arg \max }\limits_{(T_1↩
   ,T_2,T_3)} I}=$(' num2str(v(f,1)) '$,$'   num2str(v(f↩
   ,2))...
'$,$'   num2str(v(f,3)) '$) \quad p= $' num2str(num2str(↩
   p,'%.6f')) ...
'$\quad \lambda_0= $' num2str(L0)...
'$\quad  \lambda_1=$' num2str(L1)],'FontUnits','points',↩
   'FontSize',14,'Interpreter','latex');


figure
```

```
ft=14; %font size
E=[q^2  p*q p*q p^2]; Hh=-E*log2(E');
plot(v(:,3)',Hh*ones(1,length(v(:,3))),'b','LineWidth'↩
   ,2);
set(gca,'FontSize',ft);
xlabel('$T_3$','FontUnits','points','FontSize',ft,'↩
   Interpreter','latex');
ylabel('$H$','FontUnits','points','FontSize',ft,'↩
   Interpreter','latex');
hold on;

m1 = ( x'==y' ); % To find maximum only when T1=T2
m1=double(m1); % logical to double. Otherwise NaN would ↩
   not be replacable by 0 in logical m1.
m1( m1 == 0 )  =  NaN; % Replace 0 with NaNs (to ↩
   preserve the actual zeros in x )
xx=rmmissing(m1.*a); % remove NaNs
yy=rmmissing(m1.*v(:,3));  % remove NaNs
plot(yy,xx,'r','LineWidth',2);
title({['$\scriptstyle{\mathop {\arg \max }\limits_{T3} ↩
   MI} = $'...
num2str(v(f,3))  '$\quad p= $' num2str(p)...
'$\quad \lambda_0= $' num2str(L0)...
'$\quad  \lambda_1=$' num2str(L1) ...
'$\quad T=$' num2str(T)]},'FontUnits','points','FontSize↩
   ',ft,'Interpreter','latex');
h=legend('$H(X)$','$H(Y)$');
set(h,'FontUnits','points','FontSize',ft,'Interpreter','↩
   latex');
grid on
```

F.11 Code for Fig. (3.4) and Fig. (3.5). (Include code F.18.4)

```
% Gaussian Monte Carlo MI vs. T3 and Pd vs. T3 for Full ↩
   Target Support
clear all
% close all
tic
global X N C NN iter % It is introduced to count the ↩
   number of function calls.
X=1; % Total Iterations := (K^2-K)/2 * D
N=2; % Number of bins/targets
C=diag([1 1 1]);  %Covariance matrix of component ↩
   multivariate Gaussian
NN=10^7; % Monte Carlo Samples per dimension

%% Parameters
% P=1/16;
% P=1/8;
 p=0.125/8;
 L0=1;   L1=20;
 T=1; % Maximum total time
% P=1/2;
% P=1-1/16;
%  P=1-1/1024;
% q=1-P;
% T=10;     % Total time
D=400;  % Total number of data points (T1,T2,T3) for ↩
   each search for optimal value.
```

```
v=[linspace(0,T/2,D)',linspace(0,T/2,D)',linspace(T,0,D)']; % samples of (T1,T2,T3) % Constrained
% v=[linspace(0,T,D)',linspace(0,0,D)',linspace(0,0,D)']; % samples of (T1,T2,T3) % UnConstrained
 K=1; % K x K grid
 q=1-p;
iter=D;

MI=[ ];
Cd=[ ];
% Pd=[ ];


% for i=1:length(v);
% [m,h0,h1,h2,h3]=MI_2(p,L0,L1,v(i,1),v(i,2),v(i,3));  % Evaluate mutual information for each point (T1,T2,T3).
% [m,h0,h1,h2,h3]=MI_new(p,L0,L1,v(i,1),v(i,2),v(i,3)); % Evaluate mutual information for each point (T1,T2,T3).
[m,idn]=arrayfun(@Gauss_MI_2,p*ones(size(v,1),1),L0*ones(size(v,1),1),L1*ones(size(v,1),1),sqrt(v(:,1)),sqrt(v(:,2)),sqrt(v(:,3))); % It works great. Faster than for loop.

% MI_new handles the L0=0 situation
MI=cat(2,MI,m);
Cd=cat(2,Cd,idn/NN);

%%
[f1,g1]=max(MI);  % maximum value of MI
v(g1,:)

% [f2,g2]=max(Pd);   % maximum value of Pd
```

```
% v(g2,:)

[f3,g3]=max(Cd);
v(g3,:)

ft=14;  % Font size of labels + legends + axis

%%
% figure;
% plot(v(:,3),MI, 'LineWidth',2);
% set(gca,'FontSize',ft)
%  xlabel('$T_3$','FontUnits','points','FontSize',ft,'↩
   Interpreter','latex');
%  ylabel('$I(X_1,X_2;Y_1,Y_2,Y_3)$','FontUnits','points↩
   ','FontSize',ft,'Interpreter','latex');
%  title(['$\mathop {\arg \max }\limits_{T_3} I = $'...
%      num2str(v(g1,3))  '$\quad p= $' num2str(p)...
%      '$\quad \lambda_0= $' num2str(L0)...
%      '$\quad  \lambda_1=$' num2str(L1) ...
%      '$\quad T=$' num2str(T) '$\quad \mathop {\arg \↩
   max }\limits_{T_3}  C_d = $'...
%      num2str(v(g3,3)) ],'FontUnits','points','FontSize↩
   ',ft,'Interpreter','latex');
%  grid on

 %%
 figure
 E=[q^2  p*q p*q p^2]; Hh=-E*log2(E');
yyaxis left
% ax(1)=plot(v(:,3)',[  MI'  ;  Hh*ones(1,length(v(:,3))↩
   )], 'LineWidth',2);
plot(v(:,3)',[  MI'  ;  Hh*ones(1,length(v(:,3)))], '↩
   LineWidth',2);
```

```
ylabel('$I \: ,\: H$','FontUnits','points','FontSize',ft↩
   ,'Interpreter','latex'); % left y-axis
ylim([0 Hh])

yyaxis right
% ax(2)=plot(v(:,3)',Cd', 'LineWidth',2);
plot(v(:,3)',Cd', 'LineWidth',2);
ylabel('$P_d$','FontUnits','points','FontSize',ft,'↩
   Interpreter','latex'); % right y-axis
ylim([0 1])
set(gca,'FontSize',ft);
% linkaxes(ax,'x');
  title(['$\mathop {\arg \max }\limits_{T_3}  I=$'  ↩
     num2str(v(g1,3))...
    '$ \quad p= $'  num2str(p)    '$ \quad \lambda_0= $'↩
        num2str(L0)...
     '$\quad  \lambda_1=$' num2str(L1) ...
     '$\quad T=$' num2str(T)  '$\quad \mathop {\arg \max↩
         }\limits_{T_3} P_d = $'...
     num2str(v(g3,3))],'FontUnits','points','FontSize',↩
        ft,'Interpreter','latex');

xlabel('$T_3$','FontUnits','points','FontSize',ft,'↩
   Interpreter','latex');
 h=legend('$I(X_1,X_2;Y_1,Y_2,Y_3)$',['$H(X_1,X_2)=$' ↩
    num2str(Hh)],'$P_d$');
 set(h,'FontUnits','points','FontSize',ft, 'Location','↩
    SouthWest','Interpreter','latex');
 grid on
  toc
```

F.12 Code for Fig. (3.6) and Fig. (3.7). (Include code F.18.4)

```
% Mutual Information versus time a3, for Full Target Supports
% works with arrayfunc ( ). To use it for GPU

% Elapsed time is 766.434650 seconds with arrayfun ( )
% Elapsed time is  807.016181 seconds. seconds seconds with for loop
clear all
% close all

tic
% s=10^4;  % total number of samples / Data point.
% global X Umax y1 y2 y3 q

% global X K D
global X N C NN iter % It is introduced to count the number of function calls.
X=1; % Total Iterations := (K^2-K)/2 * D
N=2; % Number of bins/targets
C=diag([1 1 1]);  %Covariance matrix of component multivariate Gaussian
NN=10^4; % Monte Carlo Samples per dimension

%% Parameters

% P=1/16;
```

```
% P=1/8;
 P=0.99;
% P=1/2;
% P=1-1/16;
%  P=1-1/1024;
% q=1-P;
% T=10;      % Total time
D=400;  % Total number of data points (T1,T2,T3) for ↩
   each search for optimal value.
Tmax=1; % Maximum total time
v=[linspace(0,Tmax/2,D)',linspace(0,Tmax/2,D)',linspace(↩
   Tmax,0,D)'];  % samples of (T1,T2,T3)
% iter= length(v);

%% Grid formation
 Umax=5; % Max Limit of L1*sqrt(T)
%   Umax=20; % Max Limit of L1*T
 Umin=0; % Min Limit of L1*sqrt(T)

% K=30; % K x K grid
K=20; % K x K grid
iter= ((K^2-K)/2*D);
C1=linspace(Umin,Umax,K);
[A1,B1]=meshgrid(C1,C1); %X1 := L1.T,    X2:=L0.T

X1=A1;
Y1=B1;

X1(A1<=B1)=[ ]; % Eliminate the region (L1T<=L0T). When ↩
   element is eliminated Matlab results the row vector ↩
   instead of matrix
Y1(A1<=B1)=[ ]; % Eliminate the region (L1T<=L0T).
```

```
% X1(A1<=B1)=0; % Replace every element in matrix X1 with 0 whenever X1<=Y1
% Y1(A1<=B1)=0; % Replace every element in matrix Y1 with 0 whenever X1<=Y1

% %% my_MI_2.m grid (This grid is fixed for all values of L0 and L1). So calculated only once and for all.
% e=eps(0.5);
% y1_max=poissinv(1-e,Umax);
% y3_max=poissinv(1-e,2*Umax);
% [y1,y2,y3]=meshgrid(0:y1_max,0:y1_max,0:y3_max);

%% Computations start
T3_O=zeros(size(X1,1),size(X1,2)); % Zero Matrix containing the optimal T3 value for each pair of (L1T,L0T)
MI_O=zeros(size(X1,1),size(X1,2)); % Zero Matrix containing the optimal T3 value for each pair of (L1T,L0T)
ID_O=zeros(size(X1,1),size(X1,2)); % Zero Matrix containing the optimal T3 value for each pair of (L1T,L0T)
IDT3_O=zeros(size(X1,1),size(X1,2)); % Zero Matrix containing the optimal T3 value for each pair of (L1T,L0T)
for k=1:size(X1,1)
for l=1:size(X1,2)
% if X1(k,l)==0 % To avoid running the optimization in the region L1T<=L0T
% T3_O(k,l)=0;
% else
% Tmax=X1(k,l);
% Tmax=1; % Maximum total time
```

```
% v=[linspace(0,Tmax/2,D)',linspace(0,Tmax/2,D)',↩
   linspace(Tmax,0,D)'];  % samples of (T1,T2,T3)
% iter= length(v);
% v=[linspace(0,Tmax/2,D)',linspace(0,Tmax/2,D)',↩
   linspace(Tmax,0,D)'];  % samples of (T1,T2,T3)
% sum(v,2)
% keyboard

% if X==2
%  keyboard
% end
n=length(P); %Number of probability points
O_MI=[ ];
O_T3=[ ];
O_ID=[ ];
O_IDT3= [  ];
% O_p= [ ];
% m_info=[ ];
for j=1:n
p=P(j);    % Probability of 1 (higher rate)
% q=1-p;
% MI=[ ];

% for i=1:length(v);
% [m,h0,h1,h2,h3]=MI_new(p,Y1(k,l),X1(k,l),v(i,1),v(i,2)↩
   ,v(i,3));  % Evaluate mutual information for each ↩
   point (T1,T2,T3).
%  [m,h0,h1,h2,h3]=MI_2(p,Y1(k,l),X1(k,l),v(i,1),v(i,2),↩
   v(i,3));  % Evaluate mutual information for each point↩
    (T1,T2,T3).
% [m,h0,h1,h2,h3]=MI_2(p,Y1(k,l)/X1(k,l),1,v(i,1),v(i,2)↩
   ,v(i,3));  % Evaluate mutual information for each ↩
   point (T1,T2,T3).
```

```
% [m,h0,h1,h2,h3]=new_MI_2(p,Y1(k,l)/X1(k,l),1,v(i,1),v(i,2),v(i,3));  % Evaluate mutual information for each point (T1,T2,T3).
% [m,h0,h1,h2,h3]=My_MI(p,Y1(k,l)/X1(k,l),1,v(i,1),v(i,2),v(i,3));  % Evaluate mutual information for each point (T1,T2,T3).
% [m]=my_MI_2(p,Y1(k,l)/X1(k,l),1,v(i,1),v(i,2),v(i,3));  % Evaluate mutual information for each point (T1,T2,T3).
% [m]=my_MI_2(p,Y1(k,l),X1(k,l),v(i,1),v(i,2),v(i,3));  % Evaluate mutual information for each point (T1,T2,T3).
% m=arrayfun(@my_MI_3,gpuArray(p*ones(size(v,1),1)),gpuArray(Y1(k,l)*ones(size(v,1),1)),gpuArray(X1(k,l)*ones(size(v,1),1)),gpuArray(v(:,1)),gpuArray(v(:,2)),gpuArray(v(:,3)));
% It is under construction
% m=arrayfun(@my_MI_3,p*ones(size(v,1),1),Y1(k,l)*ones(size(v,1),1),X1(k,l)*ones(size(v,1),1),v(:,1),v(:,2),v(:,3)); % It works great. Faster than for loop.
% [m]=my_MI_3(p,Y1(k,l),X1(k,l),v(i,1),v(i,2),v(i,3));  % Evaluate mutual information for each point (T1,T2,T3).
[m,idn]=arrayfun(@Gauss_MI_2,p*ones(size(v,1),1),Y1(k,l)*ones(size(v,1),1),X1(k,l)*ones(size(v,1),1),sqrt(v(:,1)),sqrt(v(:,2)),sqrt(v(:,3))); % It works great. Faster than for loop.

% MI=cat(2,MI,m);
MI=m;
ID=idn/NN;
% end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[f1,g1]=max(MI);  % maximum value of MI
```

```
[r1,s1]=max(ID);  % maximum value of MI
% v(g1,:)
% m_info=cat(1,m_info,MI); % size: length(P) X D
O_MI=cat(2,O_MI,f1);   % Optimal MI  size: 1 X length(P)
O_T3=cat(2,O_T3,v(g1,3));    % Optimal T3 size: 1 X ↩
   length(P)
O_ID=cat(2,O_ID,r1);   % Optimal MI  size: 1 X length(P)
O_IDT3=cat(2,O_IDT3,v(s1,3));    % Optimal T3 size: 1 X ↩
   length(P)
end
T3_O(k,l)=O_T3;
MI_O(k,l)=O_MI;
ID_O(k,l)=O_ID;
IDT3_O(k,l)=O_IDT3;
% end
end
end
H=-[1-p p]*log2([1-p p]')*2; % H ( X )
 %%
% figure
% scatter(X1,Y1,[ ] ,'filled')
% ylabel('$\lambda_0 \cdot T$','Interpreter','latex');
% xlabel('$\lambda_1 \cdot T$','Interpreter','latex');
% title({['$ \rm{Grid \: for \:} \lambda_1 T> \lambda_0 ↩
   T $']},'Interpreter','latex');
% axis tight
% grid on
%%
figure
% nexttile
scatter(X1,Y1,100,MI_O ,'filled')
ylabel('$\lambda_0 $','FontSize',14,'Interpreter','latex↩
   ');
```

```
xlabel('$\lambda_1 $','FontSize',14,'Interpreter','latex');
% title({['$ p=$'...
%     num2str(p) '$\quad T=$' num2str(Tmax) '$\quad H(X)=$' num2str(H)],...
%     ['$ (T_1,T_2,T_3) := (\frac{T-\alpha}{2},\frac{T-\alpha}{2},\alpha); \quad 0 \le \alpha \le T.$']},'Interpreter','latex');

title({['$ p=$'...
    num2str(p) '$\quad H(X)=$' num2str(H)  '$\quad T=$' num2str(Tmax)]},'FontSize',14,'Interpreter','latex');

caxis([0  max(MI_O)]); % maps blue to 0 and red to maximum value
colormap(jet(256));
h=colorbar;
% h.Limits = [0  max(MI_O)];
% set(h, 'ylim', [min(MI_O) max(MI_O)])
ylabel(h,'$I^o(X;Y)$','FontUnits','points','FontSize',14,'interpreter','latex');
axis tight
grid on
%%
figure
% nexttile
% scatter(reshape(X1,1,numel(X1)),reshape(Y1,1,numel(Y1)),50,reshape(T3_O,1,numel(T3_O)),'filled')
% X1(T3_O==0)=[ ];
% Y1(T3_O==0)=[ ];
% T3_O(T3_O==0)=[ ];

scatter(X1,Y1,100,T3_O,'s','filled')
```

```
ylabel('$\lambda_0 $','FontSize',14,'Interpreter','latex↩
   ');
xlabel('$\lambda_1 $','FontSize',14,'Interpreter','latex↩
   ');
% title({['$ \rm{Optimal \: joint \: sensing \: time \: ↩
   (\alpha^o) \: in \: region \:} \lambda_1 T> \lambda_0 ↩
   T $'],...
%     ['$p=$' num2str(p) '$\quad H(X)=$' num2str(H)]},'↩
   Interpreter','latex');

title({['$p=$' num2str(p) '$\quad H(X)=$' num2str(H)  '$↩
   \quad T=$' num2str(Tmax)]},'FontSize',14,'Interpreter'↩
   ,'latex');

caxis([0  Tmax]); % maps blue to 0 and red to maximum ↩
   value
h=colorbar;
% h.Limits = [0  Tmax];
colormap(jet(256));
% h=colorbar;
% h.Limits = [0  Tmax];
% set(h, 'ylim', [min(T3_0) max(T3_0)])
ylabel(h,'$T_{3}^o$','FontUnits','points','FontSize',14,↩
   'interpreter','latex');
axis tight;
grid on;
%%
BW=T3_0;
BW(T3_0 >0 )=1;
BW(T3_0 == 0)=0;
figure
% scatter(X1,Y1,100,BW,'s','filled')
```

```
gscatter([X1 max(X1) max(X1)],[Y1 -1 -2 ],[BW  0 1],'rb',[ ],35) % Point (max(X1),-1) is included to make the extra group visible in legend
ylabel('$\lambda_0 $','FontSize',14,'Interpreter','latex');
xlabel('$\lambda_1 $','FontSize',14,'Interpreter','latex');
title({['$p=$' num2str(p) '$\quad H(X)=$' num2str(H)  '$\quad T=$' num2str(Tmax) ]},'FontSize',14,'Interpreter','latex');
% caxis([0  1]); % maps blue to 0 and red to maximum value
% h=colorbar;
% colormap(gray(2));
% ylabel(h,'$\alpha^o$','FontUnits','points','FontSize',14,'interpreter','latex');
h=legend('$T_{3}^o = 0 $','$T_{3}^o> 0$');
set(h,'FontUnits','points','FontSize',14,'Interpreter','latex');
axis tight;
axis([min(X1) max(X1) min(Y1) max(Y1)]);
grid on;
%%
cmap = colormap;
s1=length(unique(cmap,'rows')); % no. of distinct colors used in cmap
formatSpec1 = 'no. of distinct colors used in cmap:  \t\t%d  \n';
fprintf(formatSpec1,s1)
s2=length(unique(T3_0)); % no. of unique optimal T3 values
formatSpec2 = 'no. of distinct values in T3_0 : \t\t%d  \n';
fprintf(formatSpec2,s2)
```

```
s3=nnz(T3_O); % no. of non zero elements in optimal T3
formatSpec3 = 'no. of non-zero values in T3_O :       \t↩
   \t%d  \n';
fprintf(formatSpec3,s3)
s4= sum(T3_O==Tmax); % no. of ones in optimal T3
formatSpec4 = 'no. of ones in T3_O : ↩
                          \t\t%d  \n';
fprintf(formatSpec4,s4)
s5= sum(T3_O==0); % no. of zeros in optimal T3
formatSpec5 = 'no. of zeros in T3_O : ↩
                         \t\t%d  \n';
fprintf(formatSpec5,s5)
s6=numel(T3_O); % no. of zeros in optimal T3
formatSpec6 = 'no. of elements in T3_O : ↩
                    \t\t%d  \n';
fprintf(formatSpec6,s6)

%%
figure
% nexttile
scatter(X1,Y1,100,ID_O ,'filled')
ylabel('$\lambda_0 $','FontSize',14,'Interpreter','latex↩
   ');
xlabel('$\lambda_1 $','FontSize',14,'Interpreter','latex↩
   ');
% title({['$ p=$'...
%     num2str(p) '$\quad T=$' num2str(Tmax) '$\quad H(X)↩
   =$' num2str(H)],...
%     ['$ (T_1,T_2,T_3) := (\frac{T-\alpha}{2},\frac{T-\↩
   alpha}{2},\alpha); \quad 0 \le \alpha \le T.$']},'↩
   Interpreter','latex');

title({['$ p=$'...
```

```
    num2str(p) '$\quad H(X)=$' num2str(H)    '$\quad T=$'↩
        num2str(Tmax)]},'FontSize',14,'Interpreter','↩
       latex');

caxis([0  max(ID_O)]); % maps blue to 0 and red to ↩
   maximum value
colormap(jet(256));
h=colorbar;
% h.Limits = [0  max(MI_O)];
% set(h, 'ylim', [min(MI_O) max(MI_O)])
ylabel(h,'$Pd^o$','FontUnits','points','FontSize',14,'↩
   interpreter','latex');
axis tight
grid on
%%
figure
% nexttile
% scatter(reshape(X1,1,numel(X1)),reshape(Y1,1,numel(Y1)↩
   ),50,reshape(T3_O,1,numel(T3_O)),'filled')
% X1(T3_O==0)=[ ];
% Y1(T3_O==0)=[ ];
% T3_O(T3_O==0)=[ ];

scatter(X1,Y1,100,IDT3_O,'s','filled')
ylabel('$\lambda_0 $','FontSize',14,'Interpreter','latex↩
   ');
xlabel('$\lambda_1 $','FontSize',14,'Interpreter','latex↩
   ');
% title({['$ \rm{Optimal \: joint \: sensing \: time \: ↩
   (\alpha^o) \: in \: region \:} \lambda_1 T> \lambda_0 ↩
   T $'],...
%     ['$p=$' num2str(p) '$\quad H(X)=$' num2str(H)]},'↩
   Interpreter','latex');
```

```
title({['$p=$' num2str(p) '$\quad H(X)=$' num2str(H)  '$\quad T=$' num2str(Tmax)]},'FontSize',14,'Interpreter','latex');

caxis([0  Tmax]); % maps blue to 0 and red to maximum value
h=colorbar;
% h.Limits = [0  Tmax];
colormap(jet(256));
% h=colorbar;
% h.Limits = [0  Tmax];
% set(h, 'ylim', [min(T3_O) max(T3_O)])
ylabel(h,'$T_{3}^o$','FontUnits','points','FontSize',14,'interpreter','latex');
axis tight;
grid on;
%%
ID_BW=IDT3_O;
ID_BW(IDT3_O >0 )=1;
ID_BW(IDT3_O == 0)=0;
figure
% scatter(X1,Y1,100,BW,'s','filled')
gscatter([X1 max(X1) max(X1)],[Y1 -1 -2 ],[ID_BW  0 1],'rb',[ ],35) % Point (max(X1),-1) is included to make the extra group visible in legend
ylabel('$\lambda_0 $','FontSize',14,'Interpreter','latex');
xlabel('$\lambda_1 $','FontSize',14,'Interpreter','latex');
title({['$p=$' num2str(p) '$\quad H(X)=$' num2str(H)  '$\quad T=$' num2str(Tmax)]},'FontSize',14,'Interpreter','latex');
% caxis([0  1]); % maps blue to 0 and red to maximum value
```

```
% h=colorbar;
% colormap(gray(2));
% ylabel(h,'$\alpha^o$','FontUnits','points','FontSize↩
   ',14,'interpreter','latex');
h=legend('$T_{3}^o= 0 $','$T_{3}^o > 0$');
set(h,'FontUnits','points','FontSize',14,'Interpreter','↩
   latex');
axis tight;
axis([min(X1) max(X1) min(Y1) max(Y1)]);
grid on;
toc
```

F.13 Code for Fig. (4.3). (Include code F.18.7)

```
%%  Poisson MI vs. T (Pairs, triplets, individual and ↩
   Joint sensing)

clear all
%  close all

tic
global X N p L0 L1 NN Q prob
X=1;
N=4;
NN=10^7; % MonteCarlo samples
t0=1; % Total time
D=200; %no. of time divisions
p=0.125/4;
L0=0;         L1=10;
q=1-p;
```

```
%%
Q=dec2bin(0:1:2^N-1)-'2'; % Binary words
Q(Q==-2)=L0;  %Negative: to avoid failing L0=1
Q(Q==-1)=L1;
S=Q; %probabilities of each instant of X
S(S==L0)=q;
S(S==L1)=p;
prob=prod(S,2); % probability of X

%%

% v=[linspace(0,t0/2,D)',linspace(0,t0/2,D)',linspace(t0↩
   ,0,D)'];
% v=v(2:end-1,:);
%  t=linspace(0,t0,D);
%   t=[linspace(0,1,D/2) linspace(1,t0,D/2)];
t=linspace(0,t0,D);
% MI=[ ];
MI_new=[ ];
MI_monte=[ ];
MI_monte3=[ ];
MI_monte1=[ ];
MI_monte0=[ ];
Pd= [ ];
Pd3= [ ];
Pd1= [ ];
Pd0= [ ];
% H_monte=[ ];
% h_monte=[ ];
H_u=[ ];
H_u2=[ ];
H_ux=[ ];
H_ux2=[ ];
```

```
I_u=[ ];
I_u2=[ ];
FF= binopdf(0:N,N,p);  % Binomial distrubtuion
Hs=-nansum(FF.*log2(FF)); % Entropy of Binomial R.V
% for kk=1:length(t)
for kk=1:length(t)
 X
% N=4
T0 = [t(kk)/N zeros(1,N-1)]; % Individuals
T = [0 t(kk)/6 0 0]; % 6-pairs
T3 = [0 0 t(kk)/4  0]; % 4-Triplets
T1 = [zeros(1,N-1) t(kk)]; % Joint

% N=6 : [ 6   15   20   15   6   1 ]
% T0 = [t(kk)/N zeros(1,N-1)]; % Individual
% T = [0 t(kk)/15 0 0 0 0]; % 15-pairs
% T3 = [0 0 t(kk)/20  0 0 0]; % 20-Triplets
% T4 = [0 0 0 t(kk)/15 0  0]; % 15-Quadlets
% T5 = [0 0 0 0 t(kk)/6  0]; % 6-Pentalets
% T1 = [zeros(1,N-1) t(kk)]; % Joint

% N=8
% T0 = [t(kk)/N zeros(1,N-1)]; % For N=4 time elements ↩
   with time symmetry consideration   : [4 6 4 1]
%  T3 = [0 t(kk)/28 0 0 0 0 0 0];
%  T1 = [0 0 0  t(kk)/70  0 0 0 0]; % For N=8 time ↩
   elements with time symmetry consideration  : [8 28 56 ↩
   70 56 28 8 1]
%  T = [0 0 t(kk)/56 0 0 0 0 0]; % For N=4 time elements↩
    with time symmetry consideration  : [4 6 4 1]


 % T = num2cell([t(kk)/2 0]); % For N=2 time elements ↩
    with time symmetry consideration  : [2 1]
```

```
%  T3 = num2cell([0 0 t(kk)/4  0]); % For N=4 time ↩
   elements with time symmetry consideration  : [4 6 4 1]
%  T = num2cell([0 0 t(kk)/4  0]); % For N=4 time ↩
   elements with time symmetry consideration  : [4 6 4 1]
%    T = num2cell([zeros(1,N-1) t(kk)]); % For N=4 time ↩
   elements with time symmetry consideration   : [4 6 4 ↩
   1]

%  T = num2cell([t(kk)/N zeros(1,N-1)]); % For N=5 time ↩
   elements with time symmetry consideration   : [5 10 10↩
    5 1]
%  T = num2cell([0 t(kk)/10 zeros(1,3)]); % For N=5 time↩
    elements with time symmetry consideration   : [5 10 ↩
   10 5 1]
% T = num2cell([t(kk)/N zeros(1,N-1)]); % For N=6 time ↩
   elements with time symmetry consideration   : [6 15 20↩
    15 6 1]
% T = num2cell([zeros(1,N-1) t(kk) ]); % For N=6 time ↩
   elements with time symmetry consideration   : [6 15 20↩
    15 6 1]
% T = num2cell([zeros(1,4) t(kk)/6 0 ]); % For N=6 time ↩
   elements with time symmetry consideration   : [6 15 20↩
    15 6 1]
%  T = num2cell([zeros(1,2) t(kk)/20 0 0 0]); % For N=6 ↩
   time elements with time symmetry consideration   : [6 ↩
   15 20 15 6 1]
%  T = num2cell([t(kk)/N zeros(1,N-1)]); % For N=7 time ↩
   elements with time symmetry consideration   : [ 7  21 ↩
     35   35   21   7   1]
% T = num2cell([t(kk)/N zeros(1,N-1)]); % For N=10 time ↩
   elements with time symmetry consideration:  [10 45 120↩
    210 252 210 120 45 10 1]
```

```
% T = num2cell([t(kk)/N 0 ]); % Time elements with time ↩
   symmetry consideration
% T = num2cell(20*ones(1,N)); % Time elements with time ↩
   symmetry consideration
% T = num2cell(10*[1 0]); % Time elements for N=2, T1↩
   =10,T2=10,T3=0;
% T = num2cell(20*[1 1 1]); % Time elements for N=3,

% [mi]=fun_mi(p,L0,L1,t(kk)/N,0,0);  % Evaluate mutual ↩
   information for each point (T1,T2,T3).
% MI_new=cat(2,MI_new,mi);

% T=t(i);

% tt=cell2mat(T(1)); %scaling
%% Individual sensing
tt=t(kk)/N; % For individual sensing time ?
x0_max= poissinv(1-eps(0.5),L0*tt);
x0_min= poissinv(eps(0.5),L0*tt);

x1_max= poissinv(1-eps(0.5),L1*tt);
x1_min= poissinv(eps(0.5),L1*tt);

py0=poisspdf(x0_min:x0_max,L0*tt);
py1=poisspdf(x1_min:x1_max,L1*tt);

Py=q*poisspdf(min(x0_min,x1_min):max(x0_max,x1_max),L0*↩
   tt)+...
    p*poisspdf(min(x0_min,x1_min):max(x0_max,x1_max),L1*↩
       tt); %max and min are introduced
% to prevent numerical artifact
H_y=-nansum(Py.*log2(Py));
h0=-q*nansum(py0.*log2(py0));
H1=-p*nansum(py1.*log2(py1));
```

```
mi=H_y-(h0+H1);
MI_new=cat(2,MI_new,mi);

%% Combined sensing MI. T3=T;
m_u=[ ];
    L3=N*max([L0 L1])*t(kk);
%     x_min= poissinv(eps(0.5),L);
    x_max=poissinv(1-eps(0.5),L3);
for k2=0:N   %Mixture probability
    pp=poisspdf(0:x_max,((N-k2)*L0+k2*L1)*t(kk));
    pu=FF(k2+1)*pp;
    m_u=cat(1,m_u,pu);
end
ku=nansum(m_u,1);
hu=-nansum(nansum(ku.*log2(ku)));
H_u=cat(2,H_u,hu);
hux=0;
for w=0:N   % For conditional probabilities
    x_min= poissinv(eps(0.5),((N-w)*L0+w*L1)*t(kk));
    x_max=poissinv(1-eps(0.5),((N-w)*L0+w*L1)*t(kk));
    p1=poisspdf(x_min:x_max,((N-w)*L0+w*L1)*t(kk));
    hux=hux-FF(w+1)*p1*log2(p1)';
end
% keyboard
% H=nansum(nansum(m_u)) %entropy of multivariate  ↩
   poisson mixture
% H_u=cat(2,H_u,H);
H_ux=cat(2,H_ux,hux);
I=hu-hux;
I_u=cat(2,I_u,I);

%% 2 Non-overlapping Pairs sensing
% pm_u=[ ];
```

```
% c2=2; % no. of pairs count
% N2=2; % Considering a Joint sensing problem for a two-bin case
% FF2= binopdf(0:N2,N2,p);  % Binomial distrubtuion
%     L3=N2*max([L0 L1])*t(kk)/c2;
% %     x_min= poissinv(eps(0.5),L);
%     x_max=poissinv(1-eps(0.5),L3);
% for k2=0:N2   %Mixture probability
%     pp2=poisspdf(0:x_max,((N2-k2)*L0+k2*L1)*t(kk)/c2);
%     pu=FF2(k2+1)*pp2;
%     pm_u=cat(1,pm_u,pu);
% end
% ku2=nansum(pm_u,1);
% hu2=-nansum(nansum(ku2.*log2(ku2)));
% H_u2=cat(2,H_u2,hu2);
% hux2=0;
% for w2=0:N2   % For conditional probabilities
%     x_min= poissinv(eps(0.5),((N2-w2)*L0+w2*L1)*t(kk)/c2);
%     x_max=poissinv(1-eps(0.5),((N2-w2)*L0+w2*L1)*t(kk)/c2);
%     p2=poisspdf(x_min:x_max,((N2-w2)*L0+w2*L1)*t(kk)/c2);
%     hux2=hux2-FF2(w2+1)*p2*log2(p2)';
% end
% % keyboard
% % H=nansum(nansum(m_u)) %entropy of multivariate  poisson mixture
% % H_u=cat(2,H_u,H);
% H_ux2=cat(2,H_ux2,hux2);
% I2=hu2-hux2;
% I_u2=cat(2,I_u2,I2);

  %% Monte Carlo : Individual (MI vs. T and Pd vs. T)
```

```
 [info0,cd0]=Newpoiss_nn(T0);
 [~,cd0]=Newpoiss_nn(T0);
%  MI_monte0=cat(2,MI_monte0,info0);
 Pd0=cat(2,Pd0,cd0); % Total correct detections

%% Monte Carlo : 6-Pairs (MI vs. T and Pd vs. T)
[info,cd]=Newpoiss_nn(T);
 MI_monte=cat(2,MI_monte,info);
 Pd=cat(2,Pd,cd); % Total correct detections

 %% Monte Carlo : 4-Triplets (MI vs. T and Pd vs. T)
 [info3,cd3]=Newpoiss_nn(T3);
 MI_monte3=cat(2,MI_monte3,info3);
 Pd3=cat(2,Pd3,cd3); % Total correct detections

  %% Monte Carlo : Joint (MI vs. T and Pd vs. T)
 [info1,cd1]=Newpoiss_nn(T1);
 [~,cd1]=Newpoiss_nn(T1);
%  MI_monte1=cat(2,MI_monte1,info1);
 Pd1=cat(2,Pd1,cd1); % Total correct detections

X=X+1;
end


%% Figure 1 (Information)
 figure;
hold on
% plot(t,MI_monte,'LineWidth',1.5); % (Pairs,Triplets ↩
   etc) Monte Carlo based sensing
% plot(t,[0  MI_monte(2:5)  movmean(MI_monte(6:end) ,5,'↩
   Endpoints','shrink')],'LineWidth',1.5); % (Pairs,↩
   Triplets etc) Monte Carlo based sensing
```

```
% plot(t,[N*MI_new;MI_monte0], 'k','LineWidth',1.5); % Individual sensing
plot(t,N*MI_new, 'k','LineWidth',1.5); % Individual sensing
plot(t,[0 MI_monte(2:end)],'b','LineWidth',1.5); % 6-Pairs sensing
plot(t,[0 MI_monte3(2:end)],'r','LineWidth',1.5);  % 4-Triplets sensing
% plot(t,[I_u;MI_monte1],'color',[0 0.5 0],'LineWidth',1.5);  % Joint sensing (Green)
plot(t,I_u,'color',[0 0.5 0],'LineWidth',1.5);  % Joint sensing (Green)

% plot(t,c2*I_u2,'color',[0.9290 0.6940 0.1250],'LineWidth',1.5);  % 2 non-overlapping pairs (orange)
% plot(t,Pd,'color',[0 0.5 0],'LineWidth',1.5);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

hold on
HI=-N*(q*log2(q)+p*log2(p));
plot([t(1) t(end)],[HI  HI],'m','LineWidth',1)
hold on
plot([t(1) t(end)],[Hs  Hs],'c','LineWidth',1)
xlabel('$T$','FontUnits','points','FontSize',14,'interpreter','latex');
% ylabel('$N \cdot I_1(X_1;Y_1)$','FontUnits','points','FontSize',14,'interpreter','latex');
ylabel('$I$','FontUnits','points','FontSize',14,'interpreter','latex');

for s=0:N
    n(s+1)=L0*(N-s)+L1*(s);
end
```

```
%% Analytic derivatives
e3=nansum([(1-p)*(L0*log2(L0)) p*(L1*log2(L1))]);
e4=nansum([(1-p)*(L0)  p*(L1)]);
e5=nansum(e4.*log2(e4));
dd1=e3-e5 %% Analytic derivative at T=0: For T1=T2=T/2 ↩
   case
%%%%%%%%%%%%%%%
e1=nansum(FF.*(n.*log2(n)));
e2=nansum(FF.*(n))*log2(nansum(FF.*(n)));
dd2=e1-e2 %% Analytic derivative at T=0: For T=T3 case

hold on
x = [0 t0];
y = [0 dd1*t0];
line(x,y,'Color','k','LineStyle','--')

hold on
x = [0 t0];
y = [0 dd2*t0];
line(x,y,'color',[0 0.5 0],'LineStyle','--')

% hold on
% x = [0 t0];
% y = [0 L1/exp(1)/log(2)*t0]; % Maximum possible ↩
   individual-sensing rate (    L1/exp(1)    ) possible ↩
   when lambda_0=0 and lambda_1.
% line(x,y,'Color','b','LineStyle','--')
%
% hold on
% x = [0 t0];
% y = [0 0.58/log(2)*L1*t0]; % Maximum possible Joint-↩
   sensing rate possible when lambda_0=0 and lambda_1
% line(x,y,'Color','m','LineStyle','--')
% axis tight
```

```
title(['\rm{Poisson:} \quad $N= $' num2str(N) '$\quad p=↩
    $' num2str(p)...
'$\quad \lambda_0= $' num2str(L0)...
'$\quad  \lambda_1=$' num2str(L1)...
'$\quad  T=$' num2str(t0)],'FontUnits','points','↩
   FontSize',14,'Interpreter','latex');
% h=legend('$I(X_1,X_2,X_3,X_4;Y_1,Y_2,Y_3,Y_4) \quad \↩
   rm(Individual)$','$I(X_1,X_2,X_3,X_4;Y_5,Y_6,Y_7,Y_8,↩
   Y_9,Y_{10}) \quad \rm (6-pairs) $','$I(X_1,X_2,X_3,X_4↩
   ;Y_{11},Y_{12},Y_{13},Y_{14}) \rm (4-Triplets) $','$I(↩
   X_1,X_2,X_3,X_4;Y_{15}) \quad  \rm(Joint)$','$I(X_1+↩
   X_2 ,X_3+X_4;Y_5,Y_{10})=2 \cdot I(X_1+X_2 ;Y_5) \Big ↩
   |_{T=\frac{T}{2}} \quad  (2 \: \rm nonoverlapping\: ↩
   pairs)$','$H(X_1,X_2,X_3,X_4)$',['$ \frac{\partial I↩
   }{\partial T}\Big|_{T=0}=$' num2str(dd1) '$ \textrm{ (↩
   Individual-sensing)} $'],['$ \frac{\partial I}{\↩
   partial T}\Big|_{T=0}=$' num2str(dd2) '$ \textrm{ (↩
   Joint-sensing)} $'],'$\textrm{Maximum possible ↩
   theoretical individual-sensing rate $\frac{\lambda_1}{↩
   e \cdot \mathrm{log_{e}(2)}}$  at } T=0 \textrm{ and }↩
    p=\frac{1}{e}$','$\textrm{Maximum possible ↩
   theoretical Joint-sensing rate $0.58 \cdot \lambda_1 \↩
   cdot {(\mathrm{log_{e}}(2)})^{-1}$ at } T=0 \textrm{ ↩
   when } p \rightarrow 0, N \rightarrow \infty, N \cdot ↩
   p \rightarrow \lambda \approx 1.34$');
```

```
% h=legend('$I(X_1,X_2,X_3,X_4;Y_1,Y_2,Y_3,Y_4) \quad \rm(Individual)$','$I(X_1,X_2,X_3,X_4;Y_5,Y_6,Y_7,Y_8,Y_9,Y_{10}) \quad \rm (6-pairs) $','$I(X_1,X_2,X_3,X_4;Y_{11},Y_{12},Y_{13},Y_{14}) \rm (4-Triplets) $','$I(X_1,X_2,X_3,X_4;Y_{15}) \quad  \rm(Joint)$','$I(X_1+X_2 ,X_3+X_4;Y_5,Y_{10}) \quad  (2 \: \rm nonoverlapping\: pairs)$',['$H(X_1,X_2,X_3,X_4)=$' num2str(HI)] , ['$H(\sum_i X_i)=$'  num2str(Hs) ] ,['$ \frac{\partial I}{\partial T}\Big|_{T=0}=$' num2str(dd1) '$ \textrm{ (Individual-sensing)} $'],['$ \frac{\partial I}{\partial T}\Big|_{T=0}=$' num2str(dd2) '$ \textrm{ (Joint-sensing)} $']);
h=legend('$I(X_1,X_2,X_3,X_4;Y_1,Y_2,Y_3,Y_4) \quad \rm(4-Singlets)$','$I(X_1,X_2,X_3,X_4;Y_5,Y_6,Y_7,Y_8,Y_9,Y_{10}) \quad \rm (6-Pairs) $','$I(X_1,X_2,X_3,X_4;Y_{11},Y_{12},Y_{13},Y_{14}) \rm (4-Triplets) $','$I(X_1,X_2,X_3,X_4;Y_{15}) \quad  \rm(1-Quadruplet)$',['$H(X_1,X_2,X_3,X_4)=$' num2str(HI)] , ['$H(\sum_i X_i)=$' num2str(Hs) ] ,['$ \frac{\partial I}{\partial T}\Big|_{T=0}=$' num2str(dd1) '$ \textrm{ (4-Singlets)} $'],['$ \frac{\partial I}{\partial T}\Big|_{T=0}=$' num2str(dd2) '$ \textrm{ (1-Quadruplet)} $']);
set(h,'Location','southeast','color','none','NumColumns',1,'FontUnits','points','FontSize',10,'Interpreter','latex','location','best');
ylim([0 HI]);
grid on
% toc
% -2*FF2*log2(FF2') % entropy of two independent pairs
%  p=0.125;   q=1-p;   P=[q^2   2*p*q    p^2]';
% K = kron(kron(kron(kron(kron(P,P),P),P),P),P); % probabilities of all
% possible outcomes for 6 trials with each bin having 3 possible
```

```
% outcomes i-e 3^6 number of elements in support
% -sum(K.*log2(K))

%% Figure 2 (Detection)
%  figure; % WITH Moving Average
% plot(t,movmean(Pd0 ,3,'Endpoints','shrink'),'k','↩
   LineWidth',1.5); % Individual
% hold on
% plot(t,movmean(Pd ,3,'Endpoints','shrink'),'color',[0 ↩
   0.5 0],'LineWidth',1.5); %Pairs
% hold on
% plot(t,movmean(Pd3 ,3,'Endpoints','shrink'),'r','↩
   LineWidth',1.5); % Triplets
% hold on
% plot(t,movmean(Pd1,3,'Endpoints','shrink'),'b','↩
   LineWidth',1.5); % Joint

 figure; % WITHOUT Moving Average
plot(t,Pd0,'k','LineWidth',1.5); % Individual
hold on
plot(t,Pd,'b','LineWidth',1.5); %Pairs
hold on
plot(t,Pd3,'r','LineWidth',1.5); % Triplets
hold on
plot(t,Pd1,'color',[0 0.5 0],'LineWidth',1.5); % Joint


xlabel('$T$','FontUnits','points','FontSize',14,'↩
   interpreter','latex');
% ylabel('$N \cdot I_1(X_1;Y_1)$','FontUnits','points','↩
   FontSize',14,'interpreter','latex');
ylabel('$Pd$','FontUnits','points','FontSize',14,'↩
   interpreter','latex');
```

```
title(['\rm{Poisson:} \quad  $N= $' num2str(N) '$\quad p↩
   = $' num2str(p)...
'$\quad \lambda_0= $' num2str(L0)...
'$\quad  \lambda_1=$' num2str(L1)...
'$\quad  T=$' num2str(t0)],'FontUnits','points','↩
   FontSize',14,'Interpreter','latex');
h=legend('$\mathrm{4-Singlets}$','$ \mathrm{6-Pairs}$','↩
   $ \mathrm{4-Triplets}$','$ \mathrm{1-Quadruplet}$');
% h=legend('$I(X;Y) \rm (MonteCarlo: 6-pairs \: sensing)↩
    $','$I(X;Y) \rm(Computed: \: Individual-sensing)$','↩
   $I(X,Y_{2^N-1}) \: \rm(Computed: \: Joint-sensing)$','↩
   $I(X_1+X_2 ,X_3+X_4;Y_5,Y_{10})=2 \cdot I(X_1+X_2 ;Y_5↩
   ) \Big |_{T=\frac{T}{2}} $','$Pd $','$H(X_1,X_2 \cdots↩
    X_N)$','$\textrm{(Theoretical) Rate of individual-↩
   sensing curve at } T=0$','$\textrm{(Theoretical) Rate ↩
   of joint-sensing curve at } T=0 $','$\textrm{Maximum ↩
   possible theoretical individual-sensing rate $\frac{\↩
   lambda_1}{e \cdot \mathrm{log_{e}(2)}}$  at } T=0 \↩
   textrm{ and } p=\frac{1}{e}$','$\textrm{Maximum ↩
   possible theoretical Joint-sensing rate $0.58 \cdot \↩
   lambda_1 \cdot {(\mathrm{log_{e}}(2)}^{-1}$ at } T=0 ↩
   \textrm{ when } p \rightarrow 0, N \rightarrow \infty,↩
    N \cdot p \rightarrow \lambda \approx 1.34$');
% h=legend('$I(X;Y) \rm (MonteCarlo)$','$Pd $','$H(X_1,↩
   X_2 \cdots X_N)$','$\textrm{(Theoretical) Tangent to ↩
   red curve at } T=0$','$\textrm{(Theoretical) Tangent ↩
   to black curve at } T=0 $','$\textrm{(Theoretical) ↩
   Maximum possible Joint-sensing rate at } T=0 $','$\↩
   textrm{(Theoretical) Maximum possible individual-↩
   sensing rate at } T=0 $');
set(h,'Location','southeast','FontUnits','points','↩
   FontSize',14,'NumColumns',1,'Interpreter','latex','↩
   location','best');
grid on
```

```
toc
```

F.14 Code for Fig. (4.4). (Include code F.18.7)

```
%%  Poisson (Constraint) MonteCarlo for MI vs. alpha and↩
    Pd vs. alpha for full support in time constraint.
clear all
%  close all
tic
global  N NN Q prob
X=1;
N=4;
NN=10^6; % MonteCarlo samples
t0=1; % Total time
D=200; %no. of time divisions
p=0.125;
L0=2;         L1=20;
q=1-p;

%%
Q=dec2bin(0:1:2^N-1)-'2'; % Binary words
Q(Q==-2)=L0;  %Negative: to avoid failing L0=1
Q(Q==-1)=L1;
S=Q; %probabilities of each instant of X
S(S==L0)=q;
S(S==L1)=p;
prob=prod(S,2); % probability of X

%% probabilities of each word
% x=[p 1-p];
```

```
% pro=x;
% for i=1:(N-1)
% pro=kron(pro,x);
% end

%%

% v=[linspace(0,t0/2,D)',linspace(0,t0/2,D)',linspace(t0↩
   ,0,D)'];
% v=v(2:end-1,:);
alpha=linspace(0,t0,D);

% MI=[ ];
% MI_new=[ ];
MI_monte=[ ];
MI_monte1=[ ];
MI_monte3=[ ];
Pd= [ ];
Pd1= [ ];
Pd3= [ ];
% H_monte=[ ];
% h_monte=[ ];
H_u=[ ];
H_ux=[ ];
I_u=[ ];
FF= binopdf(0:N,N,p);  % Binomial distrubtuion
Hs=-nansum(FF.*log2(FF)); % Entropy of Binomial R.V
% for kk=1:length(t)
for kk=1:length(alpha)
 X
%  T = [(t0-alpha(kk))/4 0 0 alpha(kk)]; % For N=4 time ↩
   elements with time symmetry consideration  : [4 6 4 1]
```

```
T = [  (t0-alpha(kk))/4    0    0    alpha(kk)]; % For N=4 time elements with time symmetry consideration  : [4 6 4 1]
T1 = [  0     (t0-alpha(kk))/6    0      alpha(kk)]; % For N=4 time elements with time symmetry consideration  : [4 6 4 1]
T3 = [   0    0    (t0-alpha(kk))/4    alpha(kk)]; % For N=4 time elements with time symmetry consideration  : [4 6 4 1]

% T = num2cell([(t0-alpha(kk))/4 0 0 alpha(kk)]); % For N=4 time elements with time symmetry consideration  : [4 6 4 1]
%  T = num2cell([0 0 (t0-alpha(kk))/4 alpha(kk)]); % For N=4 time elements with time symmetry consideration  : [4 6 4 1]
%  T = num2cell([0 (t0-alpha(kk))/6 0 alpha(kk)]); % For N=4 time elements with time symmetry consideration  : [4 6 4 1]
%  T = num2cell([0 alpha(kk)/6 0 0]); % For N=4 time elements with time symmetry consideration  : [4 6 4 1]
%  T = num2cell([0 0 t(kk)/4  0]); % For N=4 time elements with time symmetry consideration  : [4 6 4 1]
%   T = num2cell([zeros(1,N-1) t(kk)]); % For N=4 time elements with time symmetry consideration  : [4 6 4 1]
% T = num2cell([0 0 0  t(kk)/70  0 0 0]); % For N=8 time elements with time symmetry consideration  : [8 28 56 70 56 28 1]
%  T = num2cell([t(kk)/N zeros(1,N-1)]); % For N=5 time elements with time symmetry consideration  : [5 10 10 5 1]
```

```
%  T = num2cell([0 t(kk)/10 zeros(1,3)]); % For N=5 time elements with time symmetry consideration   : [5 10 10 5 1]
% T = num2cell([t(kk)/N zeros(1,N-1)]); % For N=6 time elements with time symmetry consideration   : [6 15 20 15 6 1]
% T = num2cell([zeros(1,N-1) t(kk) ]); % For N=6 time elements with time symmetry consideration   : [6 15 20 15 6 1]
% T = num2cell([zeros(1,4) t(kk)/6 0 ]); % For N=6 time elements with time symmetry consideration   : [6 15 20 15 6 1]
%  T = num2cell([zeros(1,2) t(kk)/20 0 0 0]); % For N=6 time elements with time symmetry consideration   : [6 15 20 15 6 1]
%  T = num2cell([t(kk)/N zeros(1,N-1)]); % For N=7 time elements with time symmetry consideration   : [ 7  21  35  35  21  7  1]
% T = num2cell([t(kk)/N zeros(1,N-1)]); % For N=10 time elements with time symmetry consideration:  [10 45 120 210 252 210 120 45 10 1]


% T = num2cell([t(kk)/N 0 ]); % Time elements with time symmetry consideration
% T = num2cell(20*ones(1,N)); % Time elements with time symmetry consideration
% T = num2cell(10*[1 0]); % Time elements for N=2, T1=10,T2=10,T3=0;
% T = num2cell(20*[1 1 1]); % Time elements for N=3,

% [mi]=fun_mi(p,L0,L1,t(kk)/N,0,0);  % Evaluate mutual information for each point (T1,T2,T3).
% MI_new=cat(2,MI_new,mi);
```

```
% T=t(i);

% tt=cell2mat(T(1)); %scaling
%% H(Y|X) for arbitrary time-configurations (pairs, ↩
   triplets, etc)
% b=0;
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Q=dec2bin(0:1:2^N-1)-'2'; % Binary words
% Q(Q==-2)=L0;  %Negative: to avoid failing L0=1
% Q(Q==-1)=L1;
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% S=Q; %probabilities of each instant of X
% S(S==L0)=q;
% S(S==L1)=p;
%
% prob=prod(S,2); % probability of X
% % p_yx=0;
%  h_yx=0;
%
% mu=[ ];
% for i = 1:2^N
%
% % a=1; % Initialization
%
% M=[ ];
% h=0;
% for k=1:N
% m=T(k)*sum(nchoosek(Q(i,:),k),2);  % (2^N-1) mean ↩
   values of each Poisson mixture component per X word
% M=cat(1,M,m);
% end
% mu=cat(1,mu,M'); %Poisson mean matrix
% my_mu=mu;
```

```
% a=1;  % Initialization
% h=0;  % Initialization of conditional entropy of Y ↩
   given X
% for j=1:(2^N-1) % y1,y2,y3.....y(2^N-1)
% m1=M(j);
% m1(m1==0)=NaN; %Replace 0 with NaN to avoid -inf in ↩
   log2( )
% %%%%%%%%%%%%%%%%%%%%%%
% y_limit= poissinv(1-eps(0.5),m1);
% py_11=poisspdf(0:2*y_limit,m1);
% h1=-nansum(py_11.*log2(py_11));  %Conditional entropy ↩
   of Poisson
% %%%%%%%%%%%%%%%%%%%%%%
% % h1=0.5*log2(2*pi*exp(1)*m1);
% h1(isnan(h1))=0;
% h=h+h1; % recursive additions of (2^N-1) Gaussian ↩
   entropies. e.g %...
% % h1=(0.5*log2(2*pi*exp(1)*L0*T1)+0.5*log2(2*pi*exp(1)↩
   *L0*T2)+0.5*log2(2*pi*exp(1)*2*L0*T3));
% end
% % f=prob(i)*a;
% % f(isnan(f))=0;
% % p_yx=p_yx+f; % Conditional probability of Y given X ↩
   multiplied by P(X)
%
% g=prob(i)*h;  % p(x). H(Y|X)
% g(isnan(g))=0;
% h_yx=h_yx+g; % Conditional probability of Y given X ↩
   multiplied by P(X)
% end
%
%  %% Monte Carlo for Poisson mixture entropy H(Y)
%
% YY=[ ]; % Random samples generation
```

```
% y1=[ ]; % Output
% total_cd=0; % total correct detections from samples Y
% length_index=0; % counter of samples Y
% for j=1:2^N;
% L=my_mu(j,:);
% Z=[round(prob(j)*NN),1];
% U=repmat(L,Z);
% Y=poissrnd(U,size(U));
% ysum=zeros(size(Y,1),1);
% map=[ ]; % posterior distribution of Y-samples from ↩
   each mixture component
% for J=1:2^N;
% LL=my_mu(J,:); % 1 x 15 : row vector of my_mu
% UU=repmat(LL,Z);
% py=prob(J).*prod(poisspdf(Y,UU),2); % Posterior of ↩
   each mixture component
% %  MAP detection
% map=cat(2,map,py); % horizontal-concatenate all ↩
   posteriors of every mixture component
% ysum=ysum+py; % py evaluated at each sample Y
% end
% y1=cat(1,y1,ysum);
% [mvalue,index]=max(map,[ ], 2); % select the maxima ↩
   along every row
% % id=sum((index==j))/round(prob(j)*NN); % Total ↩
   correct detections of samples Y from every mixture ↩
   component
% % id=sum((index==j))/length(index); % Total correct ↩
   detections of samples Y from every mixture component
% id=sum((index==j)); % Total correct detections of ↩
   samples Y from every mixture component
% length_index=length_index+length(index);
% total_cd=total_cd+id;
% end
```

```
% % pause
% % keyboard
% Pd=cat(2,Pd,total_cd/length_index); % Total correct ↩
   detections
% % y1=feval(f,YY);
% y1(y1==0)=NaN; %replace 0 with NaN
% H=-nansum(log2(y1))/numel(y1); % Monte Carlo ↩
   Integration for mixture entropy
%  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% % y=pdf(gm,Z);
% % H=-nansum(log2(y))/n; % Monte Carlo Integration for ↩
   mixture entropy
% info=H-h_yx; % I(X;Y)
% MI_monte=cat(2,MI_monte,info);
[info,cd]=Newpoiss_nn(T);
MI_monte=cat(2,MI_monte,info);
Pd=cat(2,Pd,cd); % Total correct detections

%% Monte Carlo : 6-Pairs with Joint (MI vs. a and Pd vs.↩
    a)
%  [info1,cd1]=Newpoiss_nn(T1);
 [info1,cd1]=Newpoiss_nn(T1);
MI_monte1=cat(2,MI_monte1,info1);
 Pd1=cat(2,Pd1,cd1); % Total correct detections

  %% Monte Carlo : 4-Triplets with Joint (MI vs. a and ↩
     Pd vs. a)
 [info3,cd3]=Newpoiss_nn(T3);
 MI_monte3=cat(2,MI_monte3,info3);
 Pd3=cat(2,Pd3,cd3); % Total correct detections
X=X+1;
end

%% Figure 1(  I vs. a  )
```

```
 figure;
% plot(t,[MI_monte], 'LineWidth',2);
% plot(t,[MI_monte;2*MI_new], 'k','LineWidth',2);
% yyaxis left
% plot(alpha,movmean(MI_monte ,5,'Endpoints','shrink'),'↩
   LineWidth',1.5); % (Pairs,Triplets etc) Monte Carlo ↩
   based sensing

plot(alpha,MI_monte,'k','LineWidth',1.5);
hold on
plot(alpha,MI_monte1,'b','LineWidth',1.5);
hold on
plot(alpha,MI_monte3,'r','LineWidth',1.5);

% mv=movmean(MI_monte,5,'Endpoints','shrink');
% plot(alpha,[MI_monte(1)   mv(2:end)],'r','LineWidth↩
   ',1.5);
% hold on
% mv1=movmean(MI_monte1,5,'Endpoints','shrink');
% plot(alpha,[MI_monte1(1)   mv1(2:end)],'b','LineWidth↩
   ',1.5);
% hold on
% mv3=movmean(MI_monte3,5,'Endpoints','shrink');
% plot(alpha,[MI_monte3(1)   mv3(2:end)],'k','LineWidth↩
   ',1.5);

HI=-N*(q*log2(q)+p*log2(p));
hold on
plot([alpha(1) alpha(end)],[HI  HI],'m','LineWidth',1)
hold on
plot([alpha(1) alpha(end)],[Hs  Hs],'c','LineWidth',1)
% for s=0:N
%     n(s+1)=L0*(N-s)+L1*(s);
% end
```

```
ylabel('$I$','FontUnits','points','FontSize',14,'interpreter','latex');
% plot(t,N*MI_new, 'r','LineWidth',1.5); % Individual sensing
% plot(t,I_u, 'k','LineWidth',1.5);  % Joint sensing
% yyaxis right
% plot(alpha,Pd,'color',[0 0.5 0],'LineWidth',1.5);  % (Pairs,Triplets etc) Monte Carlo based sensing
% ylabel('$Pd$','FontUnits','points','FontSize',14,'interpreter','latex');
title(['\rm{Poisson:} \quad $N= $' num2str(N) '$\quad p=$' num2str(p)...
'$\quad \lambda_0= $' num2str(L0)...
'$\quad  \lambda_1=$' num2str(L1)...
'$\quad  T=$' num2str(t0)],'FontUnits','points','FontSize',14,'Interpreter','latex');
% h=legend('$I(X;Y) \rm (MonteCarlo: triples-sensing) $','$Pd $','$H(X_1,X_2 \cdots X_N)$','$\textrm{Tangent to red curve at } T=0$','$\textrm{Tangent to black curve at } T=0 $');
% set(h,'Location','southeast','FontUnits','points','FontSize',14,'Interpreter','latex','location','best');
h=legend('$ \rm ( Config-1): 4 \: singlets+1 \: quadruplet $','$  \rm ( Config-2): 6\: pairs+1 \: quadruplet $','$  \rm ( Config-3): 4 \: triplets+1 \: quadruplet $',['$H(X_1,X_2,X_3,X_4)=$' num2str(HI)] , ['$H(\sum_i X_i)=$'  num2str(Hs) ] );
set(h,'Location','southeast','FontUnits','points','FontSize',12,'Interpreter','latex','location','best');
%  xlabel('$\alpha \quad (0 \le \alpha \le T) $' ,'FontUnits','points','FontSize',14,'interpreter','latex');
 xlabel('$\alpha $' ,'FontUnits','points','FontSize',14,'interpreter','latex');
```

```
grid on
%%
% xlabel({'$$\alpha \quad 0 \le \alpha \le T $$'    , '$$↩
    \Big(\frac{T-\alpha}{4},\frac{T-\alpha}{4},\frac{T-\↩
   alpha}{4},\frac{T-\alpha}{4},0,0,0,0,0,0,0,0,0,0,\↩
   alpha \Big) $$'},'FontUnits','points','FontSize',14,'↩
   interpreter','latex');
% xlabel({'$$\alpha \quad 0 \le \alpha \le T $$'    , '$$↩
    \Big(0,0,0,0,0,0,0,0,0,0,\frac{T-\alpha}{4},\frac{T-\↩
   alpha}{4},\frac{T-\alpha}{4},\frac{T-\alpha}{4},\alpha↩
    \Big) $$'},'FontUnits','points','FontSize',14,'↩
   interpreter','latex');
% xlabel({'$$\alpha \quad 0 \le \alpha \le T $$'    , '$$↩
    \Big(0,0,0,0,\frac{T-\alpha}{6},\frac{T-\alpha}{6},\↩
   frac{T-\alpha}{6},\frac{T-\alpha}{6},\frac{T-\alpha↩
   }{6},\frac{T-\alpha}{6},0,0,0,0,\alpha \Big) $$'},'↩
   FontUnits','points','FontSize',14,'interpreter','latex↩
   ');

%% Figure 2 (Pd vs. a)
 figure;
% plot(t,[MI_monte], 'LineWidth',2);
% plot(t,[MI_monte;2*MI_new], 'k','LineWidth',2);
% yyaxis left
% plot(alpha,movmean(MI_monte ,5,'Endpoints','shrink'),'↩
   LineWidth',1.5); % (Pairs,Triplets etc) Monte Carlo ↩
   based sensing
plot(alpha,Pd,'k','LineWidth',1.5);
hold on
plot(alpha,Pd1,'b','LineWidth',1.5);
hold on
plot(alpha,Pd3,'r','LineWidth',1.5);
HI=-N*(q*log2(q)+p*log2(p));
% hold on
```

```
% plot([alpha(1) alpha(end)],[HI  HI],'m','LineWidth',1)
% hold on
% plot([alpha(1) alpha(end)],[Hs  Hs],'c','LineWidth',1)
% for s=0:N
%     n(s+1)=L0*(N-s)+L1*(s);
% end
ylabel('$P_d$','FontUnits','points','FontSize',14,'↩
   interpreter','latex');
% plot(t,N*MI_new, 'r','LineWidth',1.5); % Individual ↩
   sensing
% plot(t,I_u, 'k','LineWidth',1.5);  % Joint sensing
% yyaxis right
% plot(alpha,Pd,'color',[0 0.5 0],'LineWidth',1.5);  % (↩
   Pairs,Triplets etc) Monte Carlo based sensing
% ylabel('$Pd$','FontUnits','points','FontSize',14,'↩
   interpreter','latex');
title(['\rm{Poisson:} \quad $N= $' num2str(N) '$\quad p=↩
    $' num2str(p)...
'$\quad \lambda_0= $' num2str(L0)...
'$\quad  \lambda_1=$' num2str(L1)...
'$\quad  T=$' num2str(t0)],'FontUnits','points','↩
   FontSize',14,'Interpreter','latex');
% h=legend('$I(X;Y) \rm (MonteCarlo: triples-sensing) $↩
   ','$Pd $','$H(X_1,X_2 \cdots X_N)$','$\textrm{Tangent ↩
   to red curve at } T=0$','$\textrm{Tangent to black ↩
   curve at } T=0 $');
% set(h,'Location','southeast','FontUnits','points','↩
   FontSize',14,'Interpreter','latex','location','best');
h=legend('$  \rm ( Config-1): 4 \:singlets+1 \: ↩
   quadruplet $','$ \rm ( Config-2): 6 \:pairs+1 \:↩
   quadruplet $','$  \rm ( Config-3): 4 \: triplets+1 \:↩
   quadruplet $');
set(h,'Location','southeast','FontUnits','points','↩
   FontSize',12,'Interpreter','latex','location','best');
```

```
% xlabel('$\alpha \quad (0 \le \alpha \le T) $' ,'FontUnits','points','FontSize',14,'interpreter','latex');
xlabel('$\alpha$' ,'FontUnits','points','FontSize',14,'interpreter','latex');
grid on
toc
```

F.15 Code for Fig. (4.5). (Include code F.18.8)

```
%%  Gaussian MI vs. T (Pairs, triplets, individual and Joint sensing)

clear all
%  close all

tic
global C cov N NN Q prob
X=0;
N=4;
NN=10^6; % MonteCarlo samples
t0=1; % Total time
D=100; %no. of time divisions
p=0.5;
L0=1;         L1=10;
q=1-p;
C=eye(2^N-1);
cov=[ ]; %    2^N-1   x   2^N-1   x   2^N.  3D array containing each of component covariance matrices
for ii=1:2^N
```

```
% C=diag(mu(ii,:));
% C=eye(size(mu,2)); % Identity matrix for covariance ↩
   matrix of each multivariate Gaussian component
% C=eye(size(mu,2)); % Identity matrix for covariance ↩
   matrix of each multivariate Gaussian component
% C=diag([1 1 2]);  %Covariance matrix other than ↩
   identity matrix
cov=cat(3,cov,C);
end
% iter=D;
%%
Q=dec2bin(0:1:2^N-1)-'2'; % Binary words
Q(Q==-2)=L0;  %Negative: to avoid failing L0=1
Q(Q==-1)=L1;
S=Q; %probabilities of each instant of X
S(S==L0)=q;
S(S==L1)=p;
prob=prod(S,2); % probability of X

%%

% v=[linspace(0,t0/2,D)',linspace(0,t0/2,D)',linspace(t0↩
   ,0,D)'];
% v=v(2:end-1,:);
%  t=linspace(0,t0,D);
%   t=[linspace(0,1,D/2) linspace(1,t0,D/2)];
t=linspace(0,t0,D);
% MI=[ ];
MI_new=[ ];
MI_monte=[ ];
MI_monte3=[ ];
MI_monte1=[ ];
MI_monte0=[ ];
Pd= [ ];
```

```
Pd3= [ ];
Pd1= [ ];
Pd0= [ ];
% H_monte=[ ];
% h_monte=[ ];
H_u=[ ];
H_u2=[ ];
H_ux=[ ];
H_ux2=[ ];
I_u=[ ];
I_u2=[ ];
FF= binopdf(0:N,N,p);  % Binomial distrubtuion
Hs=-nansum(FF.*log2(FF)); % Entropy of Binomial R.V
% for kk=1:length(t)
for kk=1:length(t)
% N=4
T0 =sqrt( [t(kk)/N zeros(1,N-1)]); % For N=4 time elements with time symmetry consideration   : [4 6 4 1]
T=   sqrt( [0 t(kk)/6 0 0]); % For N=4 time elements with time symmetry consideration  : [4 6 4 1]
T3= sqrt( [0 0 t(kk)/4  0]);
T1= sqrt( [zeros(1,N-1) t(kk)]); % For N=4 time elements with time symmetry consideration   : [4 6 4 1]

% N=8
% T0 = [t(kk)/N zeros(1,N-1)]; % For N=4 time elements with time symmetry consideration   : [4 6 4 1]
%  T3 = [0 t(kk)/28 0 0 0 0 0 0];
%  T1 = [0 0 0  t(kk)/70  0 0 0 0]; % For N=8 time elements with time symmetry consideration  : [8 28 56 70 56 28 8 1]
%  T = [0 0 t(kk)/56 0 0 0 0 0]; % For N=4 time elements with time symmetry consideration  : [4 6 4 1]
```

```
 % T = num2cell([t(kk)/2 0]); % For N=2 time elements ↩
    with time symmetry consideration  : [2 1]
%  T3 = num2cell([0 0 t(kk)/4  0]); % For N=4 time ↩
   elements with time symmetry consideration  : [4 6 4 1]
%  T = num2cell([0 0 t(kk)/4  0]); % For N=4 time ↩
   elements with time symmetry consideration  : [4 6 4 1]
%    T = num2cell([zeros(1,N-1) t(kk)]); % For N=4 time ↩
   elements with time symmetry consideration   : [4 6 4 ↩
   1]

%  T = num2cell([t(kk)/N zeros(1,N-1)]); % For N=5 time ↩
   elements with time symmetry consideration   : [5 10 10↩
    5 1]
%  T = num2cell([0 t(kk)/10 zeros(1,3)]); % For N=5 time↩
    elements with time symmetry consideration   : [5 10 ↩
   10 5 1]
% T = num2cell([t(kk)/N zeros(1,N-1)]); % For N=6 time ↩
   elements with time symmetry consideration   : [6 15 20↩
    15 6 1]
% T = num2cell([zeros(1,N-1) t(kk) ]); % For N=6 time ↩
   elements with time symmetry consideration   : [6 15 20↩
    15 6 1]
% T = num2cell([zeros(1,4) t(kk)/6 0 ]); % For N=6 time ↩
   elements with time symmetry consideration   : [6 15 20↩
    15 6 1]
%  T = num2cell([zeros(1,2) t(kk)/20 0 0 0]); % For N=6 ↩
   time elements with time symmetry consideration   : [6 ↩
   15 20 15 6 1]
%  T = num2cell([t(kk)/N zeros(1,N-1)]); % For N=7 time ↩
   elements with time symmetry consideration   : [ 7  21 ↩
     35   35   21   7   1]
```

```
% T = num2cell([t(kk)/N zeros(1,N-1)]); % For N=10 time elements with time symmetry consideration:  [10 45 120 210 252 210 120 45 10 1]


% T = num2cell([t(kk)/N 0 ]); % Time elements with time symmetry consideration
% T = num2cell(20*ones(1,N)); % Time elements with time symmetry consideration
% T = num2cell(10*[1 0]); % Time elements for N=2, T1=10,T2=10,T3=0;
% T = num2cell(20*[1 1 1]); % Time elements for N=3,

% [mi]=fun_mi(p,L0,L1,t(kk)/N,0,0);  % Evaluate mutual information for each point (T1,T2,T3).
% MI_new=cat(2,MI_new,mi);

% T=t(i);

% tt=cell2mat(T(1)); %scaling
% %% Individual sensing
% tt=t(kk)/N; % For individual sensing time ?
% x0_max= poissinv(1-eps(0.5),L0*tt);
% x0_min= poissinv(eps(0.5),L0*tt);
%
% x1_max= poissinv(1-eps(0.5),L1*tt);
% x1_min= poissinv(eps(0.5),L1*tt);
%
% py0=poisspdf(x0_min:x0_max,L0*tt);
% py1=poisspdf(x1_min:x1_max,L1*tt);
%
% Py=q*poisspdf(min(x0_min,x1_min):max(x0_max,x1_max),L0*tt)+...
```

```
%     p*poisspdf(min(x0_min,x1_min):max(x0_max,x1_max),↩
   L1*tt); %max and min are introduced
% % to prevent numerical artifact
% H_y=-nansum(Py.*log2(Py));
% h0=-q*nansum(py0.*log2(py0));
% H1=-p*nansum(py1.*log2(py1));
%
% mi=H_y-(h0+H1);
% MI_new=cat(2,MI_new,mi);
%
% %% Combined sensing MI. T3=T;
% m_u=[ ];
%     L3=N*max([L0 L1])*t(kk);
% %     x_min= poissinv(eps(0.5),L);
%     x_max=poissinv(1-eps(0.5),L3);
% for k2=0:N   %Mixture probability
%     pp=poisspdf(0:x_max,((N-k2)*L0+k2*L1)*t(kk));
%     pu=FF(k2+1)*pp;
%     m_u=cat(1,m_u,pu);
% end
% ku=nansum(m_u,1);
% hu=-nansum(nansum(ku.*log2(ku)));
% H_u=cat(2,H_u,hu);
% hux=0;
% for w=0:N   % For conditional probabilities
%     x_min= poissinv(eps(0.5),((N-w)*L0+w*L1)*t(kk));
%     x_max=poissinv(1-eps(0.5),((N-w)*L0+w*L1)*t(kk));
%     p1=poisspdf(x_min:x_max,((N-w)*L0+w*L1)*t(kk));
%     hux=hux-FF(w+1)*p1*log2(p1)';
% end
% % keyboard
% % H=nansum(nansum(m_u)) %entropy of multivariate  ↩
   poisson mixture
% % H_u=cat(2,H_u,H);
```

```
% H_ux=cat(2,H_ux,hux);
% I=hu-hux;
% I_u=cat(2,I_u,I);
%
% %% Pair-sensing: 6. I(T/6) where I is a Joint sensing of two bins.
% pm_u=[ ];
% c2=2; % no. of pairs count
% N2=2; % Considering a Joint sensing problem for a two-bin case
% FF2= binopdf(0:N2,N2,p);  % Binomial distrubtuion
%     L3=N2*max([L0 L1])*t(kk)/c2;
% %     x_min= poissinv(eps(0.5),L);
%     x_max=poissinv(1-eps(0.5),L3);
% for k2=0:N2   %Mixture probability
%     pp2=poisspdf(0:x_max,((N2-k2)*L0+k2*L1)*t(kk)/c2);
%     pu=FF2(k2+1)*pp2;
%     pm_u=cat(1,pm_u,pu);
% end
% ku2=nansum(pm_u,1);
% hu2=-nansum(nansum(ku2.*log2(ku2)));
% H_u2=cat(2,H_u2,hu2);
% hux2=0;
% for w2=0:N2   % For conditional probabilities
%     x_min= poissinv(eps(0.5),((N2-w2)*L0+w2*L1)*t(kk)/c2);
%     x_max=poissinv(1-eps(0.5),((N2-w2)*L0+w2*L1)*t(kk)/c2);
%     p2=poisspdf(x_min:x_max,((N2-w2)*L0+w2*L1)*t(kk)/c2);
%     hux2=hux2-FF2(w2+1)*p2*log2(p2)';
% end
% % keyboard
```

```
% % H=nansum(nansum(m_u)) %entropy of multivariate
   poisson mixture
% % H_u=cat(2,H_u,H);
% H_ux2=cat(2,H_ux2,hux2);
% I2=hu2-hux2;
% I_u2=cat(2,I_u2,I2);

%% Monte Carlo : Individual (MI vs. T and Pd vs. T)
%  [info0,cd0]=Newpoiss_nn(T0);
 [info0,cd0]=Newgauss_nn(T0);
MI_monte0=cat(2,MI_monte0,info0);
 Pd0=cat(2,Pd0,cd0); % Total correct detections

%% Monte Carlo : 6-Pairs (MI vs. T and Pd vs. T)
[info,cd]=Newgauss_nn(T);
 MI_monte=cat(2,MI_monte,info);
 Pd=cat(2,Pd,cd); % Total correct detections

 %% Monte Carlo : 4-Triplets (MI vs. T and Pd vs. T)
 [info3,cd3]=Newgauss_nn(T3);
 MI_monte3=cat(2,MI_monte3,info3);
 Pd3=cat(2,Pd3,cd3); % Total correct detections

  %% Monte Carlo : Joint (MI vs. T and Pd vs. T)
%  [info1,cd1]=Newpoiss_nn(T1);
 [info1,cd1]=Newgauss_nn(T1);
MI_monte1=cat(2,MI_monte1,info1);
 Pd1=cat(2,Pd1,cd1); % Total correct detections

X=X+1;
PC= X/D*100;
formatSpec = 'Iteration no :  %d          Percent
   completed :  % .2f  %%      Elapsed time: % .4f  sec \
   n';
```

```
fprintf(formatSpec,X,PC,toc)

end


%% Figure 1 (Information)
 figure;
hold on
% plot(t,MI_monte,'LineWidth',1.5); % (Pairs,Triplets ↩
   etc) Monte Carlo based sensing
% plot(t,[0  MI_monte(2:5)  movmean(MI_monte(6:end) ,5,'↩
   Endpoints','shrink')],'LineWidth',1.5); % (Pairs,↩
   Triplets etc) Monte Carlo based sensing

% plot(t,N*MI_new, 'k','LineWidth',1.5); % Individual ↩
   sensing
% plot(t,[0 MI_monte(2:end)],'b','LineWidth',1.5); % 6-↩
   Pairs sensing
% plot(t,[0 MI_monte3(2:end)],'r','LineWidth',1.5);  % ↩
   4-Triplets sensing
% plot(t,I_u,'color',[0 0.5 0],'LineWidth',1.5);  % ↩
   Joint sensing
% plot(t,c2*I_u2, 'c','LineWidth',1.5);  % 6 Pair-wise ↩
   sensing caluculated from two-bin Joint sensing

plot(t,[0 MI_monte0(2:end)], 'k','LineWidth',1.5); % ↩
   Individual sensing
plot(t,[0  MI_monte(2:end)],'b','LineWidth',1.5); % 6-↩
   Pairs sensing
plot(t,[0  MI_monte3(2:end)],'r','LineWidth',1.5);  % 4-↩
   Triplets sensing
plot(t,[0  MI_monte1(2:end)],'color',[0 0.5 0],'↩
   LineWidth',1.5);  % Joint sensing
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
hold on
HI=-N*(q*log2(q)+p*log2(p));
plot([t(1) t(end)],[HI  HI],'m','LineWidth',1)
hold on
plot([t(1) t(end)],[Hs  Hs],'c','LineWidth',1)

xlabel('$T$','FontUnits','points','FontSize',14,'↩
   interpreter','latex');
% ylabel('$N \cdot I_1(X_1;Y_1)$','FontUnits','points','↩
   FontSize',14,'interpreter','latex');
ylabel('$I$','FontUnits','points','FontSize',14,'↩
   interpreter','latex');

% for s=0:N
%     n(s+1)=L0*(N-s)+L1*(s);
% end

% %% Analytic derivatives
% e3=nansum([(1-p)*(L0*log2(L0)) p*(L1*log2(L1))]);
% e4=nansum([(1-p)*(L0)  p*(L1)]);
% e5=nansum(e4.*log2(e4));
% dd1=e3-e5 %% Analytic derivative at T=0: For T1=T2=T/2↩
    case
% %%%%%%%%%%%%%%
% e1=nansum(FF.*(n.*log2(n)));
% e2=nansum(FF.*(n))*log2(nansum(FF.*(n)));
% dd2=e1-e2 %% Analytic derivative at T=0: For T=T3 case
%
% hold on
% x = [0 t0];
% y = [0 dd1*t0];
% line(x,y,'Color','k','LineStyle','--')
%
```

```
% hold on
% x = [0 t0];
% y = [0 dd2*t0];
% line(x,y,'color',[0 0.5 0],'LineStyle','--')

% hold on
% x = [0 t0];
% y = [0 L1/exp(1)/log(2)*t0]; % Maximum possible individual-sensing rate (    L1/exp(1)    ) possible when lambda_0=0 and lambda_1.
% line(x,y,'Color','b','LineStyle','--')
%
% hold on
% x = [0 t0];
% y = [0 0.58/log(2)*L1*t0]; % Maximum possible Joint-sensing rate possible when lambda_0=0 and lambda_1
% line(x,y,'Color','m','LineStyle','--')
% axis tight

title(['\rm{Gaussian:} \quad $N= $' num2str(N) '$\quad p= $' num2str(p)...
'$\quad \lambda_0= $' num2str(L0)...
'$\quad  \lambda_1=$' num2str(L1)...
'$\quad  T=$' num2str(t0)],'FontUnits','points','FontSize',14,'Interpreter','latex');
```

```
% h=legend('$I(X_1,X_2,X_3,X_4;Y_1,Y_2,Y_3,Y_4) \quad \rm(Individual)$','$I(X_1,X_2,X_3,X_4;Y_5,Y_6,Y_7,Y_8,Y_9,Y_{10}) \quad \rm (6-pairs) $','$I(X_1,X_2,X_3,X_4;Y_{11},Y_{12},Y_{13},Y_{14}) \rm (4-Triplets) $','$I(X_1,X_2,X_3,X_4;Y_{15}) \quad  \rm(Joint)$','$I(X_1+X_2 ,X_3+X_4;Y_5,Y_{10})=2 \cdot I(X_1+X_2 ;Y_5) \Big|_{T=\frac{T}{2}} \quad  (2 \: \rm nonoverlapping\: pairs)$','$H(X_1,X_2,X_3,X_4)$',['$ \frac{\partial I}{\partial T}\Big|_{T=0}=$' num2str(dd1) '$ \textrm{ (Individual-sensing)} $'],['$ \frac{\partial I}{\partial T}\Big|_{T=0}=$' num2str(dd2) '$ \textrm{ (Joint-sensing)} $'],'$\textrm{Maximum possible theoretical individual-sensing rate $\frac{\lambda_1}{e \cdot \mathrm{log_{e}(2)}}$  at } T=0 \textrm{ and } p=\frac{1}{e}$','$\textrm{Maximum possible theoretical Joint-sensing rate $0.58 \cdot \lambda_1 \cdot {(\mathrm{log_{e}}(2))^{-1}$ at } T=0 \textrm{ when } p \rightarrow 0, N \rightarrow \infty, N \cdot p \rightarrow \lambda \approx 1.34$');
h=legend('$I(X_1,X_2,X_3,X_4;Y_1,Y_2,Y_3,Y_4) \quad \rm(4-Singlets)$','$I(X_1,X_2,X_3,X_4;Y_5,Y_6,Y_7,Y_8,Y_9,Y_{10}) \quad \rm (6-Pairs) $','$I(X_1,X_2,X_3,X_4;Y_{11},Y_{12},Y_{13},Y_{14}) \rm (4-Triplets) $','$I(X_1,X_2,X_3,X_4;Y_{15}) \quad  \rm(1-Quadruplet)$',['$H(X_1,X_2,X_3,X_4)=$' num2str(HI)] , ['$H(\sum_i X_i)=$' num2str(Hs) ]);
set(h,'Location','southeast','FontUnits','points','FontSize',10,'Interpreter','latex','location','best');
ylim([0 HI]);
grid on
% toc
% -2*FF2*log2(FF2') % entropy of two independent pairs
%  p=0.125;   q=1-p;   P=[q^2   2*p*q    p^2]';
```

```
% K = kron(kron(kron(kron(kron(P,P),P),P),P),P); % ↩
   probabilities of all
% possible outcomes for 6 trials with each bin having 3 ↩
   possible
% outcomes i-e 3^6 number of elements in support
% -sum(K.*log2(K))

%% Figure 2 (Detection)
%  figure; % WITH Moving Average
% plot(t,movmean(Pd0 ,3,'Endpoints','shrink'),'k','↩
   LineWidth',1.5); % Individual
% hold on
% plot(t,movmean(Pd ,3,'Endpoints','shrink'),'color',[0 ↩
   0.5 0],'LineWidth',1.5); %Pairs
% hold on
% plot(t,movmean(Pd3 ,3,'Endpoints','shrink'),'r','↩
   LineWidth',1.5); % Triplets
% hold on
% plot(t,movmean(Pd1,3,'Endpoints','shrink'),'b','↩
   LineWidth',1.5); % Joint

 figure; % WITHOUT Moving Average
plot(t,Pd0,'k','LineWidth',1.5); % Individual
hold on
plot(t,Pd,'b','LineWidth',1.5); %Pairs
hold on
plot(t,Pd3,'r','LineWidth',1.5); % Triplets
hold on
plot(t,Pd1,'color',[0 0.5 0],'LineWidth',1.5); % Joint


xlabel('$T$','FontUnits','points','FontSize',14,'↩
   interpreter','latex');
```

```
% ylabel('$N \cdot I_1(X_1;Y_1)$','FontUnits','points','↩
   FontSize',14,'interpreter','latex');
ylabel('$Pd$','FontUnits','points','FontSize',14,'↩
   interpreter','latex');

title(['\rm{Gaussian:} \quad $N= $' num2str(N) '$\quad p↩
   = $' num2str(p)...
'$\quad \lambda_0= $' num2str(L0)...
'$\quad  \lambda_1=$' num2str(L1)...
'$\quad  T=$' num2str(t0)],'FontUnits','points','↩
   FontSize',14,'Interpreter','latex');
h=legend('$\mathrm{4-Singlets}$','$ \mathrm{ 6-Pairs}$',↩
   '$ \mathrm{ 4-Triplets}$','$ \mathrm{1-Quadruplet}$');
% h=legend('$I(X;Y) \rm (MonteCarlo: 6-pairs \: sensing)↩
    $','$I(X;Y) \rm(Computed: \: Individual-sensing)$','↩
   $I(X,Y_{2^N-1}) \: \rm(Computed: \: Joint-sensing)$','↩
   $I(X_1+X_2 ,X_3+X_4;Y_5,Y_{10})=2 \cdot I(X_1+X_2 ;Y_5↩
   ) \Big |_{T=\frac{T}{2}} $','$Pd $','$H(X_1,X_2 \cdots↩
    X_N)$','$\textrm{(Theoretical) Rate of individual-↩
   sensing curve at } T=0$','$\textrm{(Theoretical) Rate ↩
   of joint-sensing curve at } T=0 $','$\textrm{Maximum ↩
   possible theoretical individual-sensing rate $\frac{\↩
   lambda_1}{e \cdot \mathrm{log_{e}(2)}}$  at } T=0 \↩
   textrm{ and } p=\frac{1}{e}$','$\textrm{Maximum ↩
   possible theoretical Joint-sensing rate $0.58 \cdot \↩
   lambda_1 \cdot {(\mathrm{log_{e}}(2))^{-1}$ at } T=0 ↩
   \textrm{ when } p \rightarrow 0, N \rightarrow \infty,↩
    N \cdot p \rightarrow \lambda \approx 1.34$');
```

```
% h=legend('$I(X;Y) \rm (MonteCarlo)$','$Pd $','$H(X_1,↩
   X_2 \cdots X_N)$','$\textrm{(Theoretical) Tangent to ↩
   red curve at } T=0$','$\textrm{(Theoretical) Tangent ↩
   to black curve at } T=0 $','$\textrm{(Theoretical) ↩
   Maximum possible Joint-sensing rate at } T=0 $','$\↩
   textrm{(Theoretical) Maximum possible individual-↩
   sensing rate at } T=0 $');
set(h,'Location','southeast','FontUnits','points','↩
   FontSize',14,'Interpreter','latex','location','best');
grid on
toc
```

F.16 Code for Fig. (4.6). (Include code F.18.8)

```
%%  Poisson MonteCarlo for MI vs. alpha and Pd vs. alpha↩
    for full support in time constraint.


clear all
%  close all

tic
global C cov N NN Q prob
X=0;
N=4;
NN=10^6; % MonteCarlo samples
t0=1; % Total time
D=100; %no. of time divisions
p=0.125;
L0=5;         L1=10;
```

```
q=1-p;
C=eye(2^N-1);
cov=[ ]; %    2^N-1   x   2^N-1   x   2^N.  3D array  ↩
   containing each of component covariance matrices
for ii=1:2^N
% C=diag(mu(ii,:));
% C=eye(size(mu,2)); % Identity matrix for covariance ↩
   matrix of each multivariate Gaussian component
% C=eye(size(mu,2)); % Identity matrix for covariance ↩
   matrix of each multivariate Gaussian component
% C=diag([1 1 2]);  %Covariance matrix other than ↩
   identity matrix
cov=cat(3,cov,C);
end
% iter=D;
%%
Q=dec2bin(0:1:2^N-1)-'2'; % Binary words
Q(Q==-2)=L0;  %Negative: to avoid failing L0=1
Q(Q==-1)=L1;
S=Q; %probabilities of each instant of X
S(S==L0)=q;
S(S==L1)=p;
prob=prod(S,2); % probability of X

%% probabilities of each word
% x=[p 1-p];
% pro=x;
% for i=1:(N-1)
% pro=kron(pro,x);
% end

%%
```

```
% v=[linspace(0,t0/2,D)',linspace(0,t0/2,D)',linspace(t0↩
   ,0,D)'];
% v=v(2:end-1,:);
alpha=linspace(0,t0,D);

% MI=[ ];
% MI_new=[ ];
MI_monte=[ ];
 MI_monte3= [  ];
  MI_monte1= [  ];
Pd= [ ];
Pd3= [ ];
Pd1= [ ];
% H_monte=[ ];
% h_monte=[ ];
H_u=[ ];
H_ux=[ ];
I_u=[ ];
FF= binopdf(0:N,N,p);  % Binomial distrubtuion
Hs=-nansum(FF.*log2(FF)); % Entropy of Binomial R.V
% for kk=1:length(t)
for kk=1:length(alpha)
 X
 T = sqrt([  (t0-alpha(kk))/4    0    0    alpha(kk)]); ↩
    % For N=4 time elements with time symmetry ↩
    consideration  : [4 6 4 1]
 T1 = sqrt([  0     (t0-alpha(kk))/6    0      alpha(kk)↩
    ]); % For N=4 time elements with time symmetry ↩
    consideration  : [4 6 4 1]
 T3 = sqrt([   0    0    (t0-alpha(kk))/4    alpha(kk)])↩
    ; % For N=4 time elements with time symmetry ↩
    consideration  : [4 6 4 1]
```

```
%  T = num2cell([0 alpha(kk)/6 0 0]); % For N=4 time elements with time symmetry consideration  : [4 6 4 1]
%  T = num2cell([0 0 t(kk)/4  0]); % For N=4 time elements with time symmetry consideration  : [4 6 4 1]
%   T = num2cell([zeros(1,N-1) t(kk)]); % For N=4 time elements with time symmetry consideration   : [4 6 4 1]
% T = num2cell([0 0 0  t(kk)/70  0 0 0]); % For N=8 time elements with time symmetry consideration  : [8 28 56 70 56 28 1]
%  T = num2cell([t(kk)/N zeros(1,N-1)]); % For N=5 time elements with time symmetry consideration   : [5 10 10 5 1]
%  T = num2cell([0 t(kk)/10 zeros(1,3)]); % For N=5 time elements with time symmetry consideration   : [5 10 10 5 1]
% T = num2cell([t(kk)/N zeros(1,N-1)]); % For N=6 time elements with time symmetry consideration   : [6 15 20 15 6 1]
% T = num2cell([zeros(1,N-1) t(kk) ]); % For N=6 time elements with time symmetry consideration   : [6 15 20 15 6 1]
% T = num2cell([zeros(1,4) t(kk)/6 0 ]); % For N=6 time elements with time symmetry consideration   : [6 15 20 15 6 1]
%  T = num2cell([zeros(1,2) t(kk)/20 0 0 0]); % For N=6 time elements with time symmetry consideration   : [6 15 20 15 6 1]
%  T = num2cell([t(kk)/N zeros(1,N-1)]); % For N=7 time elements with time symmetry consideration   : [ 7  21  35  35  21  7  1]
% T = num2cell([t(kk)/N zeros(1,N-1)]); % For N=10 time elements with time symmetry consideration:  [10 45 120 210 252 210 120 45 10 1]
```

```
% T = num2cell([t(kk)/N 0 ]); % Time elements with time ↩
   symmetry consideration
% T = num2cell(20*ones(1,N)); % Time elements with time ↩
   symmetry consideration
% T = num2cell(10*[1 0]); % Time elements for N=2, T1↩
   =10,T2=10,T3=0;
% T = num2cell(20*[1 1 1]); % Time elements for N=3,

%% Monte Carlo : Individual with Joint (MI vs. a and Pd ↩
   vs. a)
[info,cd]=Newgauss_nn(T);
 MI_monte=cat(2,MI_monte,info);
 Pd=cat(2,Pd,cd); % Total correct detections

%% Monte Carlo : 6-Pairs with Joint (MI vs. a and Pd vs.↩
    a)
%  [info1,cd1]=Newpoiss_nn(T1);
 [info1,cd1]=Newgauss_nn(T1);
MI_monte1=cat(2,MI_monte1,info1);
 Pd1=cat(2,Pd1,cd1); % Total correct detections

  %% Monte Carlo : 4-Triplets with Joint (MI vs. a and ↩
     Pd vs. a)
 [info3,cd3]=Newgauss_nn(T3);
 MI_monte3=cat(2,MI_monte3,info3);
 Pd3=cat(2,Pd3,cd3); % Total correct detections


 X=X+1;
end

%% Figure 1(  I vs. a  )
```

```
 figure;
% plot(t,[MI_monte], 'LineWidth',2);
% plot(t,[MI_monte;2*MI_new], 'k','LineWidth',2);
% yyaxis left
% plot(alpha,movmean(MI_monte ,5,'Endpoints','shrink'),'↩
   LineWidth',1.5); % (Pairs,Triplets etc) Monte Carlo ↩
   based sensing

plot(alpha,MI_monte,'k','LineWidth',1.5);
hold on
plot(alpha,MI_monte1,'b','LineWidth',1.5);
hold on
plot(alpha,MI_monte3,'r','LineWidth',1.5);

% mv=movmean(MI_monte,5,'Endpoints','shrink');
% plot(alpha,[MI_monte(1)   mv(2:end)],'r','LineWidth↩
   ',1.5);
% hold on
% mv1=movmean(MI_monte1,5,'Endpoints','shrink');
% plot(alpha,[MI_monte1(1)   mv1(2:end)],'b','LineWidth↩
   ',1.5);
% hold on
% mv3=movmean(MI_monte3,5,'Endpoints','shrink');
% plot(alpha,[MI_monte3(1)   mv3(2:end)],'k','LineWidth↩
   ',1.5);

HI=-N*(q*log2(q)+p*log2(p));
hold on
plot([alpha(1) alpha(end)],[HI  HI],'m','LineWidth',1)
hold on
plot([alpha(1) alpha(end)],[Hs  Hs],'c','LineWidth',1)
% for s=0:N
%     n(s+1)=L0*(N-s)+L1*(s);
% end
```

```
ylabel('$I$','FontUnits','points','FontSize',14,'interpreter','latex');
% plot(t,N*MI_new, 'r','LineWidth',1.5); % Individual sensing
% plot(t,I_u, 'k','LineWidth',1.5);  % Joint sensing
% yyaxis right
% plot(alpha,Pd,'color',[0 0.5 0],'LineWidth',1.5);  % (Pairs,Triplets etc) Monte Carlo based sensing
% ylabel('$Pd$','FontUnits','points','FontSize',14,'interpreter','latex');
title(['\rm{Gaussian:} \quad $N= $' num2str(N) '$\quad p= $' num2str(p)...
'$\quad \lambda_0= $' num2str(L0)...
'$\quad  \lambda_1=$' num2str(L1)...
'$\quad  T=$' num2str(t0)],'FontUnits','points','FontSize',14,'Interpreter','latex');
% h=legend('$I(X;Y) \rm (MonteCarlo: triples-sensing) $','$Pd $','$H(X_1,X_2 \cdots X_N)$','$\textrm{Tangent to red curve at } T=0$','$\textrm{Tangent to black curve at } T=0 $');
% set(h,'Location','southeast','FontUnits','points','FontSize',14,'Interpreter','latex','location','best');
h=legend('$ \rm ( Config-1): 4 \: singlets + 1 \: quadruplet $','$  \rm ( Config-2): 6 \: pairs + 1 \: quadruplet $','$  \rm ( Config-3): 4 \: triplets + 1 \: quadruplet $',['$H(X_1,X_2,X_3,X_4)=$' num2str(HI)], ['$H(\sum_i X_i)=$'  num2str(Hs) ] );
set(h,'Location','southeast','FontUnits','points','FontSize',11,'Interpreter','latex','location','best');
%  xlabel('$\alpha \quad (0 \le \alpha \le T) $' ,'FontUnits','points','FontSize',14,'interpreter','latex');
 xlabel('$\alpha $' ,'FontUnits','points','FontSize',14,'interpreter','latex');
```

```
grid on
%%
% xlabel({'$$\alpha \quad 0 \le \alpha \le T $$'    , '$$\Big(\frac{T-\alpha}{4},\frac{T-\alpha}{4},\frac{T-\alpha}{4},\frac{T-\alpha}{4},0,0,0,0,0,0,0,0,0,\alpha \Big) $$'},'FontUnits','points','FontSize',14,'interpreter','latex');
% xlabel({'$$\alpha \quad 0 \le \alpha \le T $$'    , '$$\Big(0,0,0,0,0,0,0,0,0,\frac{T-\alpha}{4},\frac{T-\alpha}{4},\frac{T-\alpha}{4},\frac{T-\alpha}{4},\alpha \Big) $$'},'FontUnits','points','FontSize',14,'interpreter','latex');
% xlabel({'$$\alpha \quad 0 \le \alpha \le T $$'    , '$$\Big(0,0,0,0,\frac{T-\alpha}{6},\frac{T-\alpha}{6},\frac{T-\alpha}{6},\frac{T-\alpha}{6},\frac{T-\alpha}{6},\frac{T-\alpha}{6},0,0,0,0,\alpha \Big) $$'},'FontUnits','points','FontSize',14,'interpreter','latex');

%% Figure 2 (Pd vs. a)
 figure;
% plot(t,[MI_monte], 'LineWidth',2);
% plot(t,[MI_monte;2*MI_new], 'k','LineWidth',2);
% yyaxis left
% plot(alpha,movmean(MI_monte ,5,'Endpoints','shrink'),'LineWidth',1.5); % (Pairs,Triplets etc) Monte Carlo based sensing
plot(alpha,Pd,'k','LineWidth',1.5);
hold on
plot(alpha,Pd1,'b','LineWidth',1.5);
hold on
plot(alpha,Pd3,'r','LineWidth',1.5);
HI=-N*(q*log2(q)+p*log2(p));
% hold on
```

```
% plot([alpha(1) alpha(end)],[HI  HI],'m','LineWidth',1)
% hold on
% plot([alpha(1) alpha(end)],[Hs  Hs],'c','LineWidth',1)
% for s=0:N
%     n(s+1)=L0*(N-s)+L1*(s);
% end
ylabel('$P_d$','FontUnits','points','FontSize',14,'interpreter','latex');
% plot(t,N*MI_new, 'r','LineWidth',1.5); % Individual sensing
% plot(t,I_u, 'k','LineWidth',1.5);  % Joint sensing
% yyaxis right
% plot(alpha,Pd,'color',[0 0.5 0],'LineWidth',1.5);  % (Pairs,Triplets etc) Monte Carlo based sensing
% ylabel('$Pd$','FontUnits','points','FontSize',14,'interpreter','latex');
title(['\rm{Gaussian:} \quad $N= $' num2str(N) '$\quad p= $' num2str(p)...
'$\quad \lambda_0= $' num2str(L0)...
'$\quad  \lambda_1=$' num2str(L1)...
'$\quad  T=$' num2str(t0)],'FontUnits','points','FontSize',14,'Interpreter','latex');
% h=legend('$I(X;Y) \rm (MonteCarlo: triples-sensing) $','$Pd $','$H(X_1,X_2 \cdots X_N)$','$\textrm{Tangent to red curve at } T=0$','$\textrm{Tangent to black curve at } T=0 $');
% set(h,'Location','southeast','FontUnits','points','FontSize',14,'Interpreter','latex','location','best');
h=legend('$  \rm ( Config-1): 4 \: singlets+ 1 \: quadruplet$','$ \rm ( Config-2): 6 \: pairs+ 1 \: quadruplet$','$  \rm ( Config-3): 4 \:triplets+ 1 \: quadruplet$');
set(h,'Location','southeast','FontUnits','points','FontSize',14,'Interpreter','latex','location','best');
```

```
% xlabel('$\alpha \quad (0 \le \alpha \le T) $' ,'↩
   FontUnits','points','FontSize',14,'interpreter','latex↩
   ');
xlabel('$\alpha$' ,'FontUnits','points','FontSize',14,'↩
   interpreter','latex');
grid on
toc
```

F.17 Code for Fig. (E.1). (Include code F.18.3)

```
clear all
close all

%% For 1<Lambda<9 (Theorem 1)
% n=4;k=0;p=0.25; q=1-p; b=nchoosek(n,k)*p^(k)*q^(n-k)
% binopdf(0,4,p)

% L=0.025;
% m=6;
% for k=2:(2*m+1)
%  c=0;
%  for j=0:(k-1)
% d=(-1).^(k-1-j).*nchoosek(k-1,j).*log2(j+1);
% c=c+d;
%  end
%  hL(k-1)=c.*L.^k./gamma(k+1);
% %  hU(k-1)=c.*L.^k./gamma(k+1);
% end
% U=L*log2(L)-L*log2(exp(1));
% HL=nansum(hL)-U
```

```
% HU=nansum(hL)-hL(end)-U

syms L
% L=0.025;
m=6;
for k=2:(2*m+1)
 c=0;
 for j=0:(k-1)
d=(-1).^(k-1-j).*nchoosek(k-1,j).*log2(j+1);
c=c+d;
 end
 hL(k-1)=c.*L.^k./gamma(k+1);
%  hU(k-1)=c.*L.^k./gamma(k+1);
end
L=0.025:0.1:10;
for i=1:length(L)
U=L(i)*log2(L(i))-L(i)*log2(exp(1));
HL(i)=sum(subs(hL,L(i)))-U;
 HU(i)=sum(subs(hL,L(i)))-sum(subs(hL(end),L(i)))-U;
% HU(i)=sum(subs(hL,L(i)))-U;
end
% H_U=HU-HU(end);
p1=plot(L,[HL;HU],'r:','LineWidth',2)

% for i=1:length(L)
% x(i)=poissinv(1-eps(0.5),L(i));
% y=poisspdf(0:x(i),L(i));
% H(i)=-nansum(y.*log2(y));
% end
% hold on
% plot(L,H,'k','LineWidth',1)

%% For 9<Lambda (Theorem 2)
syms s L
```

```
b=0;
 for j=3:2*m+1
 x=(-1).^(j-1)*poisscentral(s,j)/(j.*(j-1).*s.^j);
 b=b+x;
 end
B=int(b,s,L,inf);

r=poisscentral(s,2*m+2)/((2*m+1)*s^(2*m+2));
R=int(r,s,L,inf);

C=0.5*log(2*pi*L)+0.5+B;
UL=-R+C;
UP=C;

L=0.1:0.1:20;
X=0;
for F=1:length(L)
G=0.5*log(2*pi*L(F))+0.5+int(b,s,L(F),inf);
lower(F)=log2(exp(1))*(-int(r,s,L(F),inf)+G);
upper(F)=log2(exp(1))*G;
X=X+1
end
% figure
hold on
p2=plot(L,lower,'b:',L,upper,'b:','LineWidth',2)
xlabel('$\lambda$','FontUnits','points','FontSize',14,'↩
   interpreter','latex');
% ylabel('$N \cdot I_1(X_1;Y_1)$','FontUnits','points','↩
   FontSize',14,'interpreter','latex');
ylabel('$H$','FontUnits','points','FontSize',14,'↩
   interpreter','latex');
title(['$\rm{Poisson \: entropy \: bounds} \quad m=$' ↩
   num2str(m)],'FontUnits','points','FontSize',14,'↩
   Interpreter','latex');
```

```
V=[0:0.1:20]; % lambda values
for i=1:length(V)
w(i)=poissinv(1-eps(0.5),V(i));
y=poisspdf(0:w(i),V(i));
HH(i)=log2(exp(1))*(-nansum(y.*log(y)));
% hh(i)=-nansum(y.*log2(y));
end
hold on
p3=plot(V,HH,'k','LineWidth',1)
% h=legend({'$\rm m = $' v,'$\rm m = $' v},'FontSize↩
   ',14,'Location','southwest','Interpreter','latex');
h=legend([p1(1) p2(1)  p3],'$ \rm{bounds \: for\: small ↩
   \: \lambda}$','$\rm{bounds \: for\: large \: \lambda}$↩
   ' ,'$ \rm{computed \: entropy}$');
set(h,'Location','southeast','FontUnits','points','↩
   FontSize',14,'Interpreter','latex','location','best');
grid on
axis([0 V(end) 0 HH(end)])
```

F.18 Supporting functions

F.18.1 Supporting function 1

```
%% Two bin simulation. Function written to work with ↩
   Two_bin.m
function[I1,I2,I3,I,Hy]=Con_MI_2(p,L0,L1,T1,T2,T3)
```

```
global X    % It is introduced to count the number of ↩
   function calls.

X
q=1-p;
% T=T1+T2+T3;
% e=eps(1);
e=eps(0.5);

y1_max= poissinv(1-e,L1*T1);
y2_max= poissinv(1-e,L1*T2);
y3_max= poissinv(1-e,(L1+L1)*T3);

[y1,y2,y3]=meshgrid(0:y1_max,0:y2_max,0:y3_max);

% Conditional Probabilities of Y1 given X
py1_00=poisspdf(y1,L0*T1);
py1_01=py1_00;
py1_10=poisspdf(y1,L1*T1);
py1_11=py1_10;

Y_1=q*py1_00+p*py1_11;  % unconditional probability of ↩
   y1

py1=Y_1(1,:,1);  %select 1st row of the first slice to ↩
   get increasing numbers of Y_1
h1=nansum(-py1.*log2(py1));
Hyx_1=nansum(-1*(q*py1_00(1,:,1).*log2(py1_00(1,:,1))+p*↩
   py1_11(1,:,1).*log2(py1_11(1,:,1))));
I1=h1-Hyx_1;

% Conditional Probabilities of Y2 given X
py2_00=poisspdf(y2,L0*T2);
py2_10=py2_00;
```

```
py2_01=poisspdf(y2,L1*T2);
py2_11=py2_01;

Y_2=q*py2_00+p*py2_11;  % unconditional probability of y2

py2=Y_2(:,1,1);   %select 1st column of the first slice to get increasing numbers of Y_2
h2=nansum(-py2.*log2(py2));
Hyx_2=nansum(-1*(q*py2_00(:,1,1).*log2(py2_00(:,1,1))+p*py2_11(:,1,1).*log2(py2_11(:,1,1))));
I2=h2-Hyx_2;

% Conditional Probabilities of Y3 given X
py3_00=poisspdf(y3,(L0+L0)*T3);
py3_01=poisspdf(y3,(L0+L1)*T3);
% py3_10=poisspdf(y3,(L1+L0)*T3);
py3_10=py3_01;
py3_11=poisspdf(y3,(L1+L1)*T3);

 py=q^2*(py1_00.*py2_00.*py3_00)  + ...
    q*p*(py1_01.*py2_01.*py3_01) +...
    p*q* (py1_10.*py2_10.*py3_10)+...
    p^2*(py1_11.*py2_11.*py3_11) ;

 h=-py.*log2(py);
 Hy=nansum(nansum(nansum(h)));

u1=py1_00.*py2_00.*py3_00; L_u1=log2(u1);
u2=py1_01.*py2_01.*py3_01; L_u2=log2(u2);
u3=py1_10.*py2_10.*py3_10; L_u3=log2(u3);
u4=py1_11.*py2_11.*py3_11; L_u4=log2(u4);

% u1(isnan(u1))=0;    u2(isnan(u2))=0;
```

```
% u3(isnan(u3))=0;    u4(isnan(u4))=0; % Logical ↩
   indexing to replace NaN with 0
% L_u1(isnan(L_u1))=0;   L_u2(isnan(L_u2))=0;
% L_u3(isnan(L_u3))=0;   L_u4(isnan(L_u4))=0; % Logical ↩
   indexing to replace NaN with 0

H00=q^2*u1;
H01=p*q*u2;
H10=p*q*u3;
H11=p^2*u4;

s=-1*(H00.*L_u1+H01.*L_u2+H10.*L_u3+H11.*L_u4);
%  s=-1*(q^2*u1.*L_u1+p*q*u2.*L_u2+p*q*u3.*L_u3+p^2*u4.*↩
   L_u4);
Hyx=nansum(nansum(nansum(s)));
Y=Y_1.*Y_2;  %% Joint probability of (y1,y2)

N_00=H00./Y;
N_01=H01./Y;
N_10=H10./Y;
N_11=H11./Y;

D_00=q^2.*py1_00.*py2_00./Y;
D_01=p*q.*py1_01.*py2_01./Y;
D_10=p*q.*py1_10.*py2_10./Y;
D_11=p^2.*py1_11.*py2_11./Y;

D=py./Y;

S1=N_00.*log2(N_00./(D_00.*D));
S2=N_01.*log2(N_01./(D_01.*D));
S3=N_10.*log2(N_10./(D_10.*D));
S4=N_11.*log2(N_11./(D_11.*D));
```

```
S=Y.*(S1+S2+S3+S4);
I3=nansum(nansum(nansum(S)));
I=Hy-Hyx;
X=X+1;
end
```

F.18.2 Supporting function 2

```
%% Two bin simulation. Function written to work with ↩
   Two_bin.m
function[I,H00,H01,H10,H11]=MI_2(p,L0,L1,T1,T2,T3)
global X    % It is introduced to count the number of ↩
   function calls.

X
q=1-p;
% T=T1+T2+T3;
% e=eps(1);
e=eps(0.5);

y1_max= poissinv(1-e,2*L1*T1);
y2_max= poissinv(1-e,2*L1*T2);
y3_max= poissinv(1-e,2*(L1+L1)*T3);

%L=95;y= poissinv(1-eps(0.5),L); z=poisscdf(y+1,L);z==1

[y1,y2,y3]=meshgrid(0:y1_max,0:y2_max,0:y3_max);

% Conditional Probabilities of Y1 given X
py1_00=poisspdf(y1,L0*T1);
py1_01=py1_00;
```

```
py1_10=poisspdf(y1,L1*T1);
py1_11=py1_10;

% Conditional Probabilities of Y2 given X
py2_00=poisspdf(y2,L0*T2);
py2_10=py2_00;
py2_01=poisspdf(y2,L1*T2);
py2_11=py2_01;

% Conditional Probabilities of Y3 given X
py3_00=poisspdf(y3,(L0+L0)*T3);
py3_01=poisspdf(y3,(L0+L1)*T3);
% py3_10=poisspdf(y3,(L1+L0)*T3);
py3_10=py3_01;
py3_11=poisspdf(y3,(L1+L1)*T3);

%  py=q^2*(py1_00.*py2_00.*py3_00)  + ...
%     q*p*(py1_01.*py2_01.*py3_01) +...
%     p*q* (py1_10.*py2_10.*py3_10)+...
%     p^2*(py1_11.*py2_11.*py3_11) ;
% % nansum(nansum(nansum(py)))
% h=-py.*log2(py);
% Hy=nansum(nansum(nansum(h)));
%
% u1=py1_00.*py2_00.*py3_00; L_u1=log2(u1);
% u2=py1_01.*py2_01.*py3_01; L_u2=log2(u2);
% u3=py1_10.*py2_10.*py3_10; L_u3=log2(u3);
% u4=py1_11.*py2_11.*py3_11; L_u4=log2(u4);
%
% % u1(isnan(u1))=0;    u2(isnan(u2))=0;
% % u3(isnan(u3))=0;    u4(isnan(u4))=0; % Logical ↩
   indexing to replace NaN with 0
% % L_u1(isnan(L_u1))=0;   L_u2(isnan(L_u2))=0;
```

```
% % L_u3(isnan(L_u3))=0;    L_u4(isnan(L_u4))=0; % ↩
   Logical indexing to replace NaN with 0
%
% H00=q^2*u1;
% H01=p*q*u2;
% H10=p*q*u3;
% H11=p^2*u4;
%
% s=-1*(H00.*L_u1+H01.*L_u2+H10.*L_u3+H11.*L_u4);
% % s=-1*(q^2*u1.*L_u1+p*q*u2.*L_u2+p*q*u3.*L_u3+p^2*u4↩
   .*L_u4);
% Hyx=nansum(nansum(nansum(s)));
%
% I=Hy-Hyx;
% X=X+1;

u1=py1_00.*py2_00.*py3_00; L_u1=log2(u1);
u2=py1_01.*py2_01.*py3_01; L_u2=log2(u2);
u3=py1_10.*py2_10.*py3_10; L_u3=log2(u3);
u4=py1_11.*py2_11.*py3_11; L_u4=log2(u4);


h00=-q^2*sum(sum(sum(u1.*L_u1,'omitnan'),'omitnan'),'↩
   omitnan');
h01=-p*q*sum(sum(sum(u2.*L_u2,'omitnan'),'omitnan'),'↩
   omitnan');
h10=-p*q*sum(sum(sum(u3.*L_u3,'omitnan'),'omitnan'),'↩
   omitnan');
h11=-p^2*sum(sum(sum(u4.*L_u4,'omitnan'),'omitnan'),'↩
   omitnan');

% s=-1*(H00.*L_u1+H01.*L_u2+H10.*L_u3+H11.*L_u4);
% s=-1*(q^2*u1.*L_u1+p*q*u2.*L_u2+p*q*u3.*L_u3+p^2*u4.*↩
   L_u4);
```

```
Hyx=h00+h01+h10+h11;

% u1 would never happen to come up with NaN. Because ↩
   Poisspdf never returns
% NaN for any of its domain. And u1 is the product of ↩
   three poisspdfs.
%
% u1(isnan(u1))=0;    u2(isnan(u2))=0;
% u3(isnan(u3))=0;    u4(isnan(u4))=0; % Logical ↩
   indexing to replace NaN with 0
%
% L_u1(isnan(L_u1))=0;   L_u2(isnan(L_u2))=0;
% L_u3(isnan(L_u3))=0;   L_u4(isnan(L_u4))=0; % Logical ↩
   indexing to replace NaN with 0

H00=q^2*u1;
H01=p*q*u2;
H10=p*q*u3;
H11=p^2*u4;


 py=q^2*(u1) + q*p*(u2) +p*q* (u3)+p^2*(u4) ;

% C=cat(4,q^2*(u1) ,q*p*(u2) ,p*q* (u3),p^2*(u4)); % ↩
   concatenation to perform element by element sum
% py=sum(C,4,'omitnan');


h=-py.*log2(py);
Hy=sum(sum(sum(h,'omitnan'),'omitnan'),'omitnan');

I=Hy-Hyx;
X=X+1;
```

```
end
```

F.18.3 Supporting function 3

```
function[mu]=poisscentral(s,n)
% n=10;
 syms s u
%  u = sym('u', [1 k]);
 u(1)=1;  u(2)=0;

 for k=2:n
      c=0;
 for j=0:k-2
x=nchoosek(k-1,j)*u(j+1);
c=c+x;
 end
%  u(3)=s*c;
 u=cat(2,u,s*c);
 end
%  u=cat(2,u,s*c);
mu=u(end);
```

F.18.4 Supporting function 4

```
%% Gaussian Monte Carlo based Two bin simulation. ↩
   Function written to work with Two_bin.m
```

```
% function[I,H00,H01,H10,H11]=Gauss_MI_2(p,L0,L1,T1,T2,↩
   T3)
function[I,idn]=Gauss_MI_2(p,L0,L1,T1,T2,T3)
% global X N C NN iter% It is introduced to count the ↩
   number of function calls.
global X N C NN iter
% X
q=1-p;
% N=2;
% C=diag([1 1 1]);  %Covariance matrix of component ↩
   multivariate Gaussian
% NN=10^6; % Samples per dimension
% e=eps(1);
%
% y1_max= poissinv(1-e,2*L1*T1);
% y2_max= poissinv(1-e,2*L1*T2);
% y3_max= poissinv(1-e,2*(L1+L1)*T3);
%
% %L=95;y= poissinv(1-eps(0.5),L); z=poisscdf(y+1,L);z↩
   ==1
%
% [y1,y2,y3]=meshgrid(0:y1_max,0:y2_max,0:y3_max);
%
% % Conditional Probabilities of Y1 given X
% py1_00=poisspdf(y1,L0*T1);
% py1_01=py1_00;
% py1_10=poisspdf(y1,L1*T1);
% py1_11=py1_10;
%
% % Conditional Probabilities of Y2 given X
% py2_00=poisspdf(y2,L0*T2);
% py2_10=py2_00;
% py2_01=poisspdf(y2,L1*T2);
% py2_11=py2_01;
```

```
%
% % Conditional Probabilities of Y3 given X
% py3_00=poisspdf(y3,(L0+L0)*T3);
% py3_01=poisspdf(y3,(L0+L1)*T3);
% py3_10=py3_01;
% py3_11=poisspdf(y3,(L1+L1)*T3);
%
% u1=py1_00.*py2_00.*py3_00; L_u1=log2(u1);
% u2=py1_01.*py2_01.*py3_01; L_u2=log2(u2);
% u3=py1_10.*py2_10.*py3_10; L_u3=log2(u3);
% u4=py1_11.*py2_11.*py3_11; L_u4=log2(u4);
%
% h00=-q^2*sum(sum(sum(u1.*L_u1,'omitnan'),'omitnan'),'↩
   omitnan');
% h01=-p*q*sum(sum(sum(u2.*L_u2,'omitnan'),'omitnan'),'↩
   omitnan');
% h10=-p*q*sum(sum(sum(u3.*L_u3,'omitnan'),'omitnan'),'↩
   omitnan');
% h11=-p^2*sum(sum(sum(u4.*L_u4,'omitnan'),'omitnan'),'↩
   omitnan');
%
% Hyx=h00+h01+h10+h11;
%
% H00=q^2*u1;
% H01=p*q*u2;
% H10=p*q*u3;
% H11=p^2*u4;
%  py=q^2*(u1) + q*p*(u2) +p*q* (u3)+p^2*(u4) ;
% h=-py.*log2(py);
% Hy=sum(sum(sum(h,'omitnan'),'omitnan'),'omitnan');
% I=Hy-Hyx;
% X=X+1;

%% Monte Carlo Gaussian MI calculations
```

```
% X
% T = num2cell([0 t(kk)/nchoosek(N,2) zeros(1,N-2)]); % ↩
   For N=4 time elements with time symmetry consideration
% T = num2cell([zeros(1,N-1) t(kk)]); % For N=4 time ↩
   elements with time symmetry consideration
% T = num2cell([t(kk)/N zeros(1,N-1)]); % For N=4 time ↩
   elements with time symmetry consideration

% T = num2cell([t(kk)/N 0 ]); % Time elements with time ↩
   symmetry consideration
% T = num2cell([v1(kk)  v3(kk)]); % Time elements with ↩
   time symmetry consideration
% T = num2cell([v1(kk)   v2(kk)  v3(kk)]); % Time ↩
   elements with time symmetry consideration
% T = [v1(kk)   v2(kk)  v3(kk)]; % Time elements with ↩
   time symmetry consideration
T = [T1 T2 T3]; % Time elements with time symmetry ↩
   consideration
% T = num2cell(20*ones(1,N)); % Time elements with time ↩
   symmetry consideration
% T = num2cell(10*[1 0]); % Time elements for N=2, T1↩
   =10,T2=10,T3=0;
% T = num2cell(20*[1 1 1]); % Time elements for N=3,
b=0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Q=dec2bin(0:1:2^N-1)-'2'; % Binary words
Q(Q==-2)=L0;  %Negative: to avoid failing L0=1
Q(Q==-1)=L1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
S=Q; %probabilities of each instant of X
S(S==L0)=q;
S(S==L1)=p;

prob=prod(S,2); % probability of X
```

```
% p_yx=0;
 h_yx=0;

mu=[ ];
for i = 1:2^N
% a=1; % Initialization
M=[ ];
h=0;
for k=1:N % This loop from 1 to N is OK when T1=T2. But it will give WRONG result if T1  !=  T2.
m=sum(nchoosek(Q(i,:),k),2);  % (2^N-1) mean values of each Gaussian component per X word
% m=T(k)*sum(nchoosek(Q(i,:),k),2);  % (2^N-1) mean values of each Gaussian component per X word
M=cat(1,M,m);
end
MM=T'.*M; % Time elements multiplied by respected intensities
mu=cat(1,mu,MM');%Gaussian mean matrix
%  my_mu=mu;
% a=1;  % Initialization
% h=0;  % Initialization of conditional entropy of Y given X
% for j=1:(2^N-1) % y1,y2,y3.....y(2^N-1)
% m1=M(j);
% m1(m1==0)=NaN; %Replace 0 with NaN to avoid -inf in log2( )
% h1=0.5*log2(2*pi*exp(1)*1);
% h1(isnan(h1))=0;
% h=h+h1; % recursive additions of (2^N-1) Gaussian entropies. e.g %...
% end
h=0.5*log2(det((2*pi*exp(1))*C));
g=prob(i)*h;  % p(x). H(Y|X)
```

```
g(isnan(g))=0;
h_yx=h_yx+g; % Conditional probability of Y given X ↩
   multiplied by P(X)


end

%%  Method 1: Monte Carlo (gmdistribution(mu,cov,pp))

% mu=mu(:,any(mu)); % removes zero columns
cov=[ ];
for ii=1:2^N
% C=diag(mu(ii,:));
% C=eye(size(mu,2)); % Identity matrix for covariance ↩
   matrix of each multivariate Gaussian component
% C=eye(size(mu,2)); % Identity matrix for covariance ↩
   matrix of each multivariate Gaussian component
% C=diag([1 1 2]);  %Covariance matrix other than ↩
   identity matrix
cov=cat(3,cov,C);
end
 gm = gmdistribution(mu,cov,prob');
%  rng('default'); % For reproducibility
 [Y,compidx] = random(gm,NN); % Generation of random ↩
    samples Y
 idx = cluster(gm,Y); % classification of random samples↩
     Y
 idn = sum((idx==compidx)); % Total no. of correct ↩
    detections

%  Z=gpuArray(Y);
y=pdf(gm,Y);
H=-nansum(log2(y))/NN; % Monte Carlo Integration for ↩
   mixture entropy
```

```
% info=gather(H)-h_yx; % I(X;Y)
I=H-h_yx; % I(X;Y)
% I=cat(2,I,info);
 %% Method 2 : Monte Carlo for Gaussian mixture
% YY=[ ]; % Random samples generation
% y1=[ ]; % Output
% for j=1:2^N;
% L=my_mu(j,:);
% Z=[round(prob(j)*NN),1];
% U=repmat(L,Z);
% 1 + 2.*randn(100,1);
% Y=U+single(randn(size(U)));
% Y=single(poissrnd(U,size(U)));
% ysum=zeros(size(Y,1),1);
% for J=1:2^N;
% LL=my_mu(J,:);
% UU=repmat(LL,Z);
% py=prob(J).*prod(normpdf(Y,UU,1),2);
% py=prob(J).*prod(poisspdf(Y,UU),2);
% ysum=ysum+py;
% end
% y1=cat(1,y1,ysum);
% end
% y1(y1==0)=NaN; %replace 0 with NaN
% H=-nansum(log2(y1))/numel(y1); % Monte Carlo ↩
   Integration for mixture entropy
% info=H-h_yx; % I(X;Y)
% MI=cat(2,MI,info);
% [X X/(K^2-K)/2*D*100] % Iteration no   and  Percent ↩
   simulation completed
PC= X/iter*100;
formatSpec = 'Iteration no :  %d         Percent ↩
   completed :  % .2f  %%      Elapsed time: % .4f  sec \↩
   n';
```

```
fprintf(formatSpec,X,PC,toc)
X=X+1;
end
```

F.18.5 Supporting function 5

```
%% Gaussian Monte Carlo based Two bin simulation. ↩
   Function written to work with Two_bin.m
% function[I,H00,H01,H10,H11]=Gauss_MI_2(p,L0,L1,T1,T2,↩
   T3)
function[I]=Gauss_MI(p,L0,L1,T1,T2,T3)
% global X N C NN iter% It is introduced to count the ↩
   number of function calls.
global X N C NN iter
% X
q=1-p;
% N=2;
% C=diag([1 1 1]);  %Covariance matrix of component ↩
   multivariate Gaussian
% NN=10^6; % Samples per dimension
% e=eps(1);
%
% y1_max= poissinv(1-e,2*L1*T1);
% y2_max= poissinv(1-e,2*L1*T2);
% y3_max= poissinv(1-e,2*(L1+L1)*T3);
%
% %L=95;y= poissinv(1-eps(0.5),L); z=poisscdf(y+1,L);z↩
   ==1
%
% [y1,y2,y3]=meshgrid(0:y1_max,0:y2_max,0:y3_max);
%
```

```
% % Conditional Probabilities of Y1 given X
% py1_00=poisspdf(y1,L0*T1);
% py1_01=py1_00;
% py1_10=poisspdf(y1,L1*T1);
% py1_11=py1_10;
%
% % Conditional Probabilities of Y2 given X
% py2_00=poisspdf(y2,L0*T2);
% py2_10=py2_00;
% py2_01=poisspdf(y2,L1*T2);
% py2_11=py2_01;
%
% % Conditional Probabilities of Y3 given X
% py3_00=poisspdf(y3,(L0+L0)*T3);
% py3_01=poisspdf(y3,(L0+L1)*T3);
% py3_10=py3_01;
% py3_11=poisspdf(y3,(L1+L1)*T3);
%
% u1=py1_00.*py2_00.*py3_00; L_u1=log2(u1);
% u2=py1_01.*py2_01.*py3_01; L_u2=log2(u2);
% u3=py1_10.*py2_10.*py3_10; L_u3=log2(u3);
% u4=py1_11.*py2_11.*py3_11; L_u4=log2(u4);
%
% h00=-q^2*sum(sum(sum(u1.*L_u1,'omitnan'),'omitnan'),'↩
   omitnan');
% h01=-p*q*sum(sum(sum(u2.*L_u2,'omitnan'),'omitnan'),'↩
   omitnan');
% h10=-p*q*sum(sum(sum(u3.*L_u3,'omitnan'),'omitnan'),'↩
   omitnan');
% h11=-p^2*sum(sum(sum(u4.*L_u4,'omitnan'),'omitnan'),'↩
   omitnan');
%
% Hyx=h00+h01+h10+h11;
%
```

```
% H00=q^2*u1;
% H01=p*q*u2;
% H10=p*q*u3;
% H11=p^2*u4;
%  py=q^2*(u1) + q*p*(u2) +p*q* (u3)+p^2*(u4) ;
% h=-py.*log2(py);
% Hy=sum(sum(sum(h,'omitnan'),'omitnan'),'omitnan');
% I=Hy-Hyx;
% X=X+1;

%% Monte Carlo Gaussian MI calculations
% X
% T = num2cell([0 t(kk)/nchoosek(N,2) zeros(1,N-2)]); % For N=4 time elements with time symmetry consideration
% T = num2cell([zeros(1,N-1) t(kk)]); % For N=4 time elements with time symmetry consideration
% T = num2cell([t(kk)/N zeros(1,N-1)]); % For N=4 time elements with time symmetry consideration

% T = num2cell([t(kk)/N 0 ]); % Time elements with time symmetry consideration
% T = num2cell([v1(kk)  v3(kk)]); % Time elements with time symmetry consideration
% T = num2cell([v1(kk)   v2(kk)  v3(kk)]); % Time elements with time symmetry consideration
% T = [v1(kk)   v2(kk)  v3(kk)]; % Time elements with time symmetry consideration
T = [T1 T2 T3]; % Time elements with time symmetry consideration
% T = num2cell(20*ones(1,N)); % Time elements with time symmetry consideration
% T = num2cell(10*[1 0]); % Time elements for N=2, T1=10,T2=10,T3=0;
% T = num2cell(20*[1 1 1]); % Time elements for N=3,
```

```
b=0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Q=dec2bin(0:1:2^N-1)-'2'; % Binary words
Q(Q==-2)=L0;  %Negative: to avoid failing L0=1
Q(Q==-1)=L1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
S=Q; %probabilities of each instant of X
S(S==L0)=q;
S(S==L1)=p;

prob=prod(S,2); % probability of X
% p_yx=0;
 h_yx=0;

mu=[ ];
for i = 1:2^N
% a=1; % Initialization
M=[ ];
h=0;
for k=1:N % This loop from 1 to N is OK when T1=T2. But it will give WRONG result if T1  !=  T2.
m=sum(nchoosek(Q(i,:),k),2);  % (2^N-1) mean values of each Gaussian component per X word
% m=T(k)*sum(nchoosek(Q(i,:),k),2);  % (2^N-1) mean values of each Gaussian component per X word
M=cat(1,M,m);
end
MM=T'.*M; % Time elements multiplied by respected intensities
mu=cat(1,mu,MM');%Gaussian mean matrix
%  my_mu=mu;
% a=1;  % Initialization
% h=0;  % Initialization of conditional entropy of Y given X
```

```
% for j=1:(2^N-1) % y1,y2,y3.....y(2^N-1)
% m1=M(j);
% m1(m1==0)=NaN; %Replace 0 with NaN to avoid -inf in log2( )
% h1=0.5*log2(2*pi*exp(1)*1);
% h1(isnan(h1))=0;
% h=h+h1; % recursive additions of (2^N-1) Gaussian entropies. e.g %...
% end
h=0.5*log2(det((2*pi*exp(1))*C));
g=prob(i)*h;  % p(x). H(Y|X)
g(isnan(g))=0;
h_yx=h_yx+g; % Conditional probability of Y given X multiplied by P(X)


end

%%  Method 1: Monte Carlo (gmdistribution(mu,cov,pp))

% mu=mu(:,any(mu)); % removes zero columns
cov=[ ];
for ii=1:2^N
% C=diag(mu(ii,:));
% C=eye(size(mu,2)); % Identity matrix for covariance matrix of each multivariate Gaussian component
% C=eye(size(mu,2)); % Identity matrix for covariance matrix of each multivariate Gaussian component
% C=diag([1 1 2]);  %Covariance matrix other than identity matrix
cov=cat(3,cov,C);
end
 gm = gmdistribution(mu,cov,prob');
%  rng('default'); % For reproducibility
```

```
 [Y] = random(gm,NN); % Generation of random samples Y
%  idx = cluster(gm,Y); % classification of random ↩
   samples Y
%  idn = sum((idx==compidx)); % Total no. of correct ↩
   detections

%  Z=gpuArray(Y);
y=pdf(gm,Y);
H=-nansum(log2(y))/NN; % Monte Carlo Integration for ↩
   mixture entropy
% info=gather(H)-h_yx; % I(X;Y)
I=H-h_yx; % I(X;Y)
% I=cat(2,I,info);
 %% Method 2 : Monte Carlo for Gaussian mixture
% YY=[ ]; % Random samples generation
% y1=[ ]; % Output
% for j=1:2^N;
% L=my_mu(j,:);
% Z=[round(prob(j)*NN),1];
% U=repmat(L,Z);
% 1 + 2.*randn(100,1);
% Y=U+single(randn(size(U)));
% Y=single(poissrnd(U,size(U)));
% ysum=zeros(size(Y,1),1);
% for J=1:2^N;
% LL=my_mu(J,:);
% UU=repmat(LL,Z);
% py=prob(J).*prod(normpdf(Y,UU,1),2);
% py=prob(J).*prod(poisspdf(Y,UU),2);
% ysum=ysum+py;
% end
% y1=cat(1,y1,ysum);
% end
% y1(y1==0)=NaN; %replace 0 with NaN
```

```
% H=-nansum(log2(y1))/numel(y1); % Monte Carlo ↩
   Integration for mixture entropy
% info=H-h_yx; % I(X;Y)
% MI=cat(2,MI,info);
% [X X/(K^2-K)/2*D*100] % Iteration no   and  Percent ↩
   simulation completed
PC= X/iter*100;
formatSpec = 'Iteration no :  %d          Percent ↩
   completed :  % .2f  %%      Elapsed time: % .4f  sec \↩
   n';
fprintf(formatSpec,X,PC,toc)
X=X+1;
end
```

F.18.6 Supporting function 6

```
%% Two bin simulation. Function written to work with ↩
   Two_bin.m
%% This function handles well L0=0 situation then its ↩
   counterpart MI_2.m.
function[I]=my_MI_3(p,L0,L1,T1,T2,T3)
global X K D;

% X
q=1-p;
e=eps(0.5);
y1_max=poissinv(1-e,L1*T1);
y3_max=poissinv(1-e,2*L1*T3);
% y1_max=gpuArray(200);
% y3_max=gpuArray(200);
[y1,y2,y3]=meshgrid(0:y1_max,0:y1_max,0:y3_max);
```

```
% It is introduced to count the number of function calls↩
    .

% [X X/(K^2-K)/2*D*100] % Iteration no   and  Percent ↩
    simulation completed
C= X/((K^2-K)/2*D)*100;
formatSpec = 'Iteration no :  %d          Percent ↩
    completed :  % .2f  %%      Elapsed time: % .4f  sec \↩
    n';
fprintf(formatSpec,X,C,toc)

% Conditional Probabilities of Y1 given X
py1_00=poisspdf(y1,L0*T1);
py1_01=py1_00;
py1_10=poisspdf(y1,L1*T1);
py1_11=py1_10;

% Conditional Probabilities of Y2 given X
py2_00=poisspdf(y2,L0*T2);
py2_10=py2_00;
py2_01=poisspdf(y2,L1*T2);
py2_11=py2_01;

% Conditional Probabilities of Y3 given X
py3_00=poisspdf(y3,(L0+L0)*T3);
py3_01=poisspdf(y3,(L0+L1)*T3);
% py3_10=poisspdf(y3,(L1+L0)*T3);
py3_10=py3_01;
py3_11=poisspdf(y3,(L1+L1)*T3);

%  py=q^2*(py1_00.*py2_00.*py3_00)  + ...
%     q*p*(py1_01.*py2_01.*py3_01) +...
%     p*q* (py1_10.*py2_10.*py3_10)+...
%     p^2*(py1_11.*py2_11.*py3_11) ;
```

```
% h=-py.*log2(py);
% Hy=nansum(nansum(nansum(h)));

u1=py1_00.*py2_00.*py3_00; L_u1=log2(u1);
u2=py1_01.*py2_01.*py3_01; L_u2=log2(u2);
u3=py1_10.*py2_10.*py3_10; L_u3=log2(u3);
u4=py1_11.*py2_11.*py3_11; L_u4=log2(u4);

% E1=nansum(nansum(nansum(y1.*y2.*y3.*u4))) % expected ↩
   value of variable
% (Y1*Y2*Y3|L1,L1)
% ee1=T1*L1*T2*L1*(L1+L1)*T3% expected value

H00=-q^2*sum(sum(sum(u1.*L_u1,'omitnan'),'omitnan'),'↩
   omitnan');
H01=-p*q*sum(sum(sum(u2.*L_u2,'omitnan'),'omitnan'),'↩
   omitnan');
H10=-p*q*sum(sum(sum(u3.*L_u3,'omitnan'),'omitnan'),'↩
   omitnan');
H11=-p^2*sum(sum(sum(u4.*L_u4,'omitnan'),'omitnan'),'↩
   omitnan');

% s=-1*(H00.*L_u1+H01.*L_u2+H10.*L_u3+H11.*L_u4);
% s=-1*(q^2*u1.*L_u1+p*q*u2.*L_u2+p*q*u3.*L_u3+p^2*u4.*↩
   L_u4);
Hyx=H00+H01+H10+H11;

% u1(isnan(u1))=0;    u2(isnan(u2))=0;
% u3(isnan(u3))=0;    u4(isnan(u4))=0; % Logical ↩
   indexing to replace NaN with 0
% L_u1(isnan(L_u1))=0;   L_u2(isnan(L_u2))=0;
% L_u3(isnan(L_u3))=0;   L_u4(isnan(L_u4))=0; % Logical ↩
   indexing to replace NaN with 0
```

```
 py=q^2*(u1) + q*p*(u2) +p*q* (u3)+p^2*(u4) ;
%  py=nansum([q^2*(u1)  q*p*(u2)   p*q* (u3)   p^2*(u4)↩
   ]);
%  E2=p^2*nansum(nansum(nansum(y1.*y2.*y3.*u1))) +...
%      p*q*nansum(nansum(nansum(y1.*y2.*y3.*u2)))+...
%      p*q*nansum(nansum(nansum(y1.*y2.*y3.*u3))) +...
%      q^2*nansum(nansum(nansum(y1.*y2.*y3.*u4))) % ↩
   expected value of variable

h=-py.*log2(py);
Hy=sum(sum(sum(h,'omitnan'),'omitnan'),'omitnan');

I=Hy-Hyx;
X=X+1;


end
```

F.18.7 Supporting function 7

```
%% For general N - Poisson-MonteCarlo (This is modified ↩
   from poiss_nn.m to output Pd too along with MI)
% There is no p, L0 and L1 required in this function.
function[info,Pd]=Newpoiss_nn(v)

global  N NN Q prob
% T0 nck % In workspace it would be initialize for ↩
   counting iterations. By:   >> global X; X=0;

%%
```

```
% run the following in workspace:
% global X N p L0 L1 n T0 nck
% X=0;   N=4;    p=0.25;    L0=2;        L1=4;        NN=10^5;        T0=5;
% nck=[ ]; for i=1:N
% y=nchoosek(N,i);
% nck=cat(2,nck,y);
% end
%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%For optimtool: no. of variables:    N   Aeq:  nck
          beq: T0     Bounds lower:   zeros(1,N)
   Bounds upper: T0*ones(1,N)./nck

% q=1-p;
% u=[ ];
% for bb=1:N
% z=nchoosek(N,bb);
% u=cat(2,u,z);
% end
 % Samples per dimension

% I_N([0 0 10 0 0 0]./u)
% T = num2cell(v); % Time elements with time symmetry consideration
T=v;
% T = num2cell(10*[1 0]); % Time elements for N=2, T1=10,T2=10,T3=0;
% T = num2cell(20*[1 1 1]); % Time elements for N=3,


% p_yx=0;
 h_yx=0;
```

```
mu=[ ];
for i = 1:2^N

% a=1; % Initialization

M=[ ];
h=0;
for k=1:N
m=T(k)*sum(nchoosek(Q(i,:),k),2);  % (2^N-1) mean values of each Poisson component per X word
M=cat(1,M,m);
end
mu=cat(1,mu,M'); % mean matrix
my_mu=mu;
% a=1;  % Initialization
h=0;  % Initialization of conditional entropy of Y given X
for j=1:(2^N-1) % y1,y2,y3.....y(2^N-1)
m1=M(j);
m1(m1==0)=NaN; %Replace 0 with NaN to avoid -inf in log2( )
%%%%%%%%%%%%%%%%%%%%%%
y_limit= poissinv(1-eps(0.5),m1);
py_11=poisspdf(0:2*y_limit,m1);
h1=-nansum(py_11.*log2(py_11));  %Conditional entropy of Poisson
%%%%%%%%%%%%%%%%%%%%%%
% h1=0.5*log2(2*pi*exp(1)*m1);
h1(isnan(h1))=0;
h=h+h1; % recursive additions of (2^N-1) Gaussian entropies. e.g %...
% h1=(0.5*log2(2*pi*exp(1)*L0*T1)+0.5*log2(2*pi*exp(1)*L0*T2)+0.5*log2(2*pi*exp(1)*2*L0*T3));
```

```
end
% f=prob(i)*a;
% f(isnan(f))=0;
% p_yx=p_yx+f; % Conditional probability of Y given X ↩
   multiplied by P(X)

g=prob(i)*h;  % p(x). H(Y|X)
g(isnan(g))=0;
h_yx=h_yx+g; % Conditional probability of Y given X ↩
   multiplied by P(X)
end

 %% Poisson mixture for Monte Carlo

% YY=[ ]; % Random samples generation
y1=[ ]; % Output
total_cd=0; % total correct detections from samples Y
length_index=0; % counter of samples Y
for j=1:2^N;
L=my_mu(j,:);
Z=[round(prob(j)*NN),1];
U=repmat(L,Z);
Y=poissrnd(U,size(U));
ysum=zeros(size(Y,1),1);
 map=[ ]; % posterior distribution of Y-samples from ↩
    each mixture component
for J=1:2^N;
LL=my_mu(J,:);
UU=repmat(LL,Z);
py=prob(J).*prod(poisspdf(Y,UU),2);
%  MAP detection
map=cat(2,map,py); % horizontal-concatenate all ↩
   posteriors of every mixture component
ysum=ysum+py;
```

```
end
y1=cat(1,y1,ysum);
% [mvalue,index]=max(map,[ ], 2); % select the maxima ↩
   along every row
[~,index]=max(map,[ ], 2); % select the maxima along ↩
   every row
% id=sum((index==j))/round(prob(j)*NN); % Total correct ↩
   detections of samples Y from every mixture component
% id=sum((index==j))/length(index); % Total correct ↩
   detections of samples Y from every mixture component
id=sum((index==j)); % Total correct detections of ↩
   samples Y from every mixture component
length_index=length_index+length(index);
total_cd=total_cd+id;
end
% pause
% keyboard
Pd=total_cd/length_index; % Total correct detections
% y1=feval(f,YY);
% y1=feval(f,YY);
y1(y1==0)=NaN; %replace 0 with NaN
H=-nansum(log2(y1))/numel(y1); % Monte Carlo Integration↩
    for mixture entropy
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% y=pdf(gm,Z);
% H=-nansum(log2(y))/n; % Monte Carlo Integration for ↩
   mixture entropy
info=H-h_yx; % I(X;Y)
% X=X+1
end

% H_monte=cat(2,H_monte,H);
% h_monte=cat(2,h_monte,h_yx);
% info=H-h_yx % I(X;Y)
```

```
% H_x=-prob'*log2(prob) % H(X)
```

F.18.8 Supporting function 8

```
%% Gaussian Monte Carlo based Two bin simulation. ↩
   Function written to work with Two_bin.m
% function[I,H00,H01,H10,H11]=Gauss_MI_2(p,L0,L1,T1,T2,↩
   T3)
function[I,idn]=Newgauss_nn(v)
% global X N C NN iter% It is introduced to count the ↩
   number of function calls.
global C cov N NN Q prob

% X
% q=1-p;
% N=2;
% C=diag([1 1 1]);  %Covariance matrix of component ↩
   multivariate Gaussian
% NN=10^6; % Samples per dimension
% e=eps(1);
%
% y1_max= poissinv(1-e,2*L1*T1);
% y2_max= poissinv(1-e,2*L1*T2);
% y3_max= poissinv(1-e,2*(L1+L1)*T3);
%
% %L=95;y= poissinv(1-eps(0.5),L); z=poisscdf(y+1,L);z↩
   ==1
%
% [y1,y2,y3]=meshgrid(0:y1_max,0:y2_max,0:y3_max);
%
% % Conditional Probabilities of Y1 given X
```

```
% py1_00=poisspdf(y1,L0*T1);
% py1_01=py1_00;
% py1_10=poisspdf(y1,L1*T1);
% py1_11=py1_10;
%
% % Conditional Probabilities of Y2 given X
% py2_00=poisspdf(y2,L0*T2);
% py2_10=py2_00;
% py2_01=poisspdf(y2,L1*T2);
% py2_11=py2_01;
%
% % Conditional Probabilities of Y3 given X
% py3_00=poisspdf(y3,(L0+L0)*T3);
% py3_01=poisspdf(y3,(L0+L1)*T3);
% py3_10=py3_01;
% py3_11=poisspdf(y3,(L1+L1)*T3);
%
% u1=py1_00.*py2_00.*py3_00; L_u1=log2(u1);
% u2=py1_01.*py2_01.*py3_01; L_u2=log2(u2);
% u3=py1_10.*py2_10.*py3_10; L_u3=log2(u3);
% u4=py1_11.*py2_11.*py3_11; L_u4=log2(u4);
%
% h00=-q^2*sum(sum(sum(u1.*L_u1,'omitnan'),'omitnan'),'↩
   omitnan');
% h01=-p*q*sum(sum(sum(u2.*L_u2,'omitnan'),'omitnan'),'↩
   omitnan');
% h10=-p*q*sum(sum(sum(u3.*L_u3,'omitnan'),'omitnan'),'↩
   omitnan');
% h11=-p^2*sum(sum(sum(u4.*L_u4,'omitnan'),'omitnan'),'↩
   omitnan');
%
% Hyx=h00+h01+h10+h11;
%
% H00=q^2*u1;
```

```
% H01=p*q*u2;
% H10=p*q*u3;
% H11=p^2*u4;
%  py=q^2*(u1) + q*p*(u2) +p*q* (u3)+p^2*(u4) ;
% h=-py.*log2(py);
% Hy=sum(sum(sum(h,'omitnan'),'omitnan'),'omitnan');
% I=Hy-Hyx;
% X=X+1;

%% Monte Carlo Gaussian MI calculations
% X
% T = num2cell([0 t(kk)/nchoosek(N,2) zeros(1,N-2)]); % For N=4 time elements with time symmetry consideration
% T = num2cell([zeros(1,N-1) t(kk)]); % For N=4 time elements with time symmetry consideration
% T = num2cell([t(kk)/N zeros(1,N-1)]); % For N=4 time elements with time symmetry consideration

% T = num2cell([t(kk)/N 0 ]); % Time elements with time symmetry consideration
% T = num2cell([v1(kk)  v3(kk)]); % Time elements with time symmetry consideration
% T = num2cell([v1(kk)   v2(kk)  v3(kk)]); % Time elements with time symmetry consideration
% T = [v1(kk)   v2(kk)  v3(kk)]; % Time elements with time symmetry consideration
T=v;
% T = [T1 T2 T3]; % Time elements with time symmetry consideration
% T = num2cell(20*ones(1,N)); % Time elements with time symmetry consideration
% T = num2cell(10*[1 0]); % Time elements for N=2, T1=10,T2=10,T3=0;
% T = num2cell(20*[1 1 1]); % Time elements for N=3,
```

```
% b=0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Q=dec2bin(0:1:2^N-1)-'2'; % Binary words
% Q(Q==-2)=L0;  %Negative: to avoid failing L0=1
% Q(Q==-1)=L1;
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% S=Q; %probabilities of each instant of X
% S(S==L0)=q;
% S(S==L1)=p;
%
% prob=prod(S,2); % probability of X
% p_yx=0;
 h_yx=0;

mu=[ ];
for i = 1:2^N
% a=1; % Initialization
M=[ ];
h=0;
for k=1:N % This loop from 1 to N is OK when T1=T2. But it will give WRONG result if T1  !=  T2.
m=T(k)*sum(nchoosek(Q(i,:),k),2);  % (2^N-1) mean values of each Poisson component per X word
% m=sum(nchoosek(Q(i,:),k),2);  % Enable it for unequal T1 and T2 and T3
M=cat(1,M,m);
end
% MM=T'.*M; %  Enable it  for unequal times T1 and T2
% mu=cat(1,mu,MM'); % Enable it  for unequal times T1 and T2
mu=cat(1,mu,M');%Gaussian  mean matrix
%  my_mu=mu;
% a=1;  % Initialization
```

```
% h=0;  % Initialization of conditional entropy of Y ↩
   given X
% for j=1:(2^N-1) % y1,y2,y3.....y(2^N-1)
% m1=M(j);
% m1(m1==0)=NaN; %Replace 0 with NaN to avoid -inf in ↩
   log2( )
% h1=0.5*log2(2*pi*exp(1)*1);
% h1(isnan(h1))=0;
% h=h+h1; % recursive additions of (2^N-1) Gaussian ↩
   entropies. e.g %...
% end
h=0.5*log2(det((2*pi*exp(1))*C));
g=prob(i)*h;  % p(x). H(Y|X)
g(isnan(g))=0;
h_yx=h_yx+g; % Conditional probability of Y given X ↩
   multiplied by P(X)


end

%%  Method 1: Monte Carlo (gmdistribution(mu,cov,pp))

% mu=mu(:,any(mu)); % removes zero columns
% cov=[ ];
% for ii=1:2^N
% % C=diag(mu(ii,:));
% % C=eye(size(mu,2)); % Identity matrix for covariance ↩
   matrix of each multivariate Gaussian component
% % C=eye(size(mu,2)); % Identity matrix for covariance ↩
   matrix of each multivariate Gaussian component
% % C=diag([1 1 2]);  %Covariance matrix other than ↩
   identity matrix
% cov=cat(3,cov,C);
% end
```

```
 gm = gmdistribution(mu,cov,prob');
%  rng('default'); % For reproducibility
 [Y,compidx] = random(gm,NN); % Generation of random samples Y
 idx = cluster(gm,Y); % classification of random samples Y
 idn = sum((idx==compidx))/NN; % Total no. of correct detections

%  Z=gpuArray(Y);
y=pdf(gm,Y);
H=-nansum(log2(y))/NN; % Monte Carlo Integration for mixture entropy
% info=gather(H)-h_yx; % I(X;Y)
I=H-h_yx; % I(X;Y)
% I=cat(2,I,info);
 %% Method 2 : Monte Carlo for Gaussian mixture
% YY=[ ]; % Random samples generation
% y1=[ ]; % Output
% for j=1:2^N;
% L=my_mu(j,:);
% Z=[round(prob(j)*NN),1];
% U=repmat(L,Z);
% 1 + 2.*randn(100,1);
% Y=U+single(randn(size(U)));
% Y=single(poissrnd(U,size(U)));
% ysum=zeros(size(Y,1),1);
% for J=1:2^N;
% LL=my_mu(J,:);
% UU=repmat(LL,Z);
% py=prob(J).*prod(normpdf(Y,UU,1),2);
% py=prob(J).*prod(poisspdf(Y,UU),2);
% ysum=ysum+py;
% end
```

```
% y1=cat(1,y1,ysum);
% end
% y1(y1==0)=NaN; %replace 0 with NaN
% H=-nansum(log2(y1))/numel(y1); % Monte Carlo ↩
   Integration for mixture entropy
% info=H-h_yx; % I(X;Y)
% MI=cat(2,MI,info);
% [X X/(K^2-K)/2*D*100] % Iteration no   and  Percent ↩
   simulation completed

% PC= X/iter*100;
% formatSpec = 'Iteration no :  %d          Percent ↩
   completed :  % .2f  %%      Elapsed time: % .4f  sec \↩
   n';
% fprintf(formatSpec,X,PC,toc)
% X=X+1;
end
```