*Supported by the
National Science Foundation
Engineering Research Center
Program (NSFD CD 8803012),
Industry and the University*

User's Guide for FSQP Version 1.0B A Fortran Software for Solving Optimization Problems with General Inequality Constraints and Linear Equality Constraints, Generating Feasible Iterates

by J. Zhou and A.L. Tits

TR 89-61r1

User's Guide for FSQP Version 1.0B
A Fortran Software for Solving Optimization Problems with General Inequality Constraints and Linear Equality Constraints, Generating Feasible Iterates[1]

Jian Zhou and André L. Tits

Electrical Engineering Department

and

Systems Research Center

University of Maryland, College Park, MD 20742

Abstract

FSQP is a set of Fortran subroutines for the minimization of a smooth objective function subject to nonlinear smooth inequality constraints, linear inequality and linear equality constraints, and simple bounds on the variables. If the initial guess provided by the user is infeasible, FSQP first generates a feasible point from the given point. Subsequently the successive iterates generated by FSQP all satisfy the constraints. The user also has the option of requiring that the objective value decrease at each iteration after feasibility has been reached. The user must provide subroutines that define the objective and constraint functions and may either provide the subroutines that define the gradients of these functions or require that FSQP estimate them by forward finite differences.

FSQP uses an algorithm based on Sequential Quadratic Programming (SQP), modified so as to generate feasible iterates. A certain arc search ensures that the step of one is eventually satisfied, a requirement for superlinear convergence. The merit function used in this arc search is the objective function itself, and either an Armijo-type line search or a nonmonotone line search borrowed from Grippo *et al.* may be selected.

[1]This research was supported in part by NSF's Engineering Research Centers Program No. NSFD-CDR-88-03012, by NSF grant No. DMC-88-15996 and by a grant from the Westinghouse Corporation

Contents

1 Introduction

FSQP (Feasible Sequential Quadratic Programming) is a set of Fortran subroutines for the minimization of a smooth objective function subject to nonlinear inequality constraints, linear inequality and equality constraints, and simple bounds on the variables. If the initial guess provided by the user is infeasible, FSQP first generates a feasible point from the given point. Subsequently the successive iterates generated by FSQP all satisfy the constraints. The user also has the option of requiring that the objective value decrease at each iteration after feasibility has been reached. The user must provide subroutines that define the objective and constraint functions and may either provide the subroutines that define the gradients of these functions or require that FSQP estimate them by forward finite differences.

FSQP implements an algorithm that follows the method described and analyzed in [1], with some additional refinements. This method is based on a Sequential Quadratic Programming (SQP) iteration, modified so as to generate feasible iterates. The SQP direction is first "tilted" to yield a feasible direction, then "bent" to ensure that close to a solution the step of one is accepted, a requirement for superlinear convergence. The merit function used in searching along the resulting arc is the objective function itself, and either an Armijo-type line search or a nonmonotone line search borrowed from Grippo *et al.* [2] may be selected.

FSQP invokes the quadratic programming routine QPSOL [3] which the user is supposed to provide.

2 Description of the Algorithm

The algorithm described and analyzed in [1] is as follows. Given a feasible iterate x, the basic SQP direction d^0 is first computed by solving a standard quadratic program using a positive definite estimate H of the Hessian of the Lagrangian. d^0 is a direction of descent for the objective function and is almost feasible in the sense that it is at worst tangent to the feasible set. Next an essentially arbitrary feasible descent direction $d^1 = d^1(x)$ is computed and, for a certain scalar $\rho = \rho(x) \in (0,1]$, a feasible descent search direction $d = (1-\rho)d^0 + \rho d^1$ is obtained, asymptotically close to d^0. Finally a correction $\tilde{d} = \tilde{d}(x,H)$ is computed, and a search is performed along the arc $x + td + t^2\tilde{d}$. The purpose of $\tilde{d}$ is to allow a full step of one to be taken close to a solution, thus allowing superlinear convergence to take place. Conditions are given in [1] on $d^1(\cdot)$, $\rho(\cdot)$, and $\tilde{d}(\cdot,\cdot)$ that result in a globally convergent, locally superlinear convergent algorithm.

The FSQP implementation corresponds to a specific choice of these functions, with some modifications as follows. First d^1 is computed as a function not only of x but also of d^0 (thus of H), as it appears beneficial to keep d^1 relatively close to d^0. The analysis in [1] can be easily extended to this case. Second, obvious simplifications are introduced concerning the linear inequality constraints, and linear equality constraints are allowed as well. The iterates are allowed (resp. forced) to stay on the boundary of these constraints and these constraints are not checked in the line search. Finally, FSQP automatically switches to a "phase 1" mode if the initial guess provided by the user is not in the feasible set.

Specifically, FSQP is designed to solve problems of form:

$$\begin{array}{ll} \min\limits_{x\in R^n} & f(x) \\ \text{s.t.} & bl \le x \le bu \\ & g_j(x) \le 0, \quad j=1,\dots,n_i \\ & g_j(x) \equiv \langle c_{j-n_i}, x\rangle - d_{j-n_i} \le 0, \quad j=n_i+1,\dots,t_i \\ & \langle a_j, x\rangle = b_j \quad j=1,\dots,l_e \end{array}$$

with $bl \in R^n$; $bu \in R^n$; $a_j \in R^n$, $b_j \in R$, $j=1,\dots,l_e$; $f: R^n \to R$ smooth; $g_j: R^n \to R$, $j = 1,\dots,n_i$ smooth; $c_j \in R^n$, $d_j \in R$, $j=1,\dots,t_i-n_i$.

Algorithm FSQP.

Data. $x_0 \in R^n$, $\varepsilon > 0$.

Parameters. $\eta = 0.1$, $\nu = 0.01$, $\alpha = 1.0 \times 10^{-7}$, $\beta = 0.8$.

Step 0: Initialization. Set $k=0$ and $H_0 = I$, the identity matrix. If x_0 is infeasible, find a feasible point as discussed below.

Step 1: Computation of a search arc.

i. Compute d_k^0 by solving the strictly convex quadratic program

$$\begin{array}{ll} \min\limits_{d^0\in R^n} & \frac{1}{2}\langle d^0, H_k d^0\rangle + \langle \nabla f(x_k), d^0\rangle \\ \text{s.t.} & bl \le x_k + d^0 \le bu \\ & g_j(x_k) + \langle \nabla g_j(x_k), d^0\rangle \le 0, \quad j=1,\dots,t_i \\ & \langle a_j, x_k + d^0\rangle = b_j, \quad j=1,\dots,l_e \end{array}$$

Compute the Kuhn-Tucker vector

$$\nabla L(x_k,\xi_k,\lambda_k,\mu_k) = \nabla f(x_k) + \sum_{j=1}^{n} \xi_{k,j} + \sum_{j=1}^{t_i} \lambda_{k,j}\nabla g_j(x_k) + \sum_{j=1}^{l_e} \mu_{k,j} a_j$$

where the $\xi_{k,j}$'s, $\lambda_{k,j}$'s, and $\mu_{k,j}$'s are the multipliers, for simple bounds (only n possible active bounds at each iteration), inequality, and equality constraints respectively, associated with this quadratic program.

If $\|\nabla L(x_k, \xi_k, \lambda_k, \mu_k)\| \leq \varepsilon$, stop.

ii. Compute d_k^1 by solving the strictly convex quadratic program

$$\begin{array}{ll} \min\limits_{d^1\in R^n,\gamma\in R} & \frac{\eta}{2}\langle d_k^0 - d^1, d_k^0 - d^1\rangle + \gamma \\ \text{s.t.} & bl \leq x_k + d^1 \leq bu \\ & \langle \nabla f(x_k), d^1\rangle \leq \gamma \\ & g_j(x_k) + \langle \nabla g_j(x_k), d^1\rangle \leq \gamma, \quad j = 1,\ldots,n_i \\ & \langle c_j, x_k + d^1\rangle \leq d_j, \quad j = 1,\ldots,t_i - n_i \\ & \langle a_j, x_k + d^1\rangle = b_j, \quad j = 1,\ldots,l_e \end{array}$$

iii. Set $d_k = (1-\rho_k)d_k^0 + \rho_k d_k^1$ with $\rho_k = \|d_k^0\|^{2.1}/(\|d_k^0\|^{2.1} + v_k)$, where $v_k = \max(0.5, \|d_k^1\|^{2.5})$.

iv. Compute $\tilde{d}_k$ by solving the strictly convex quadratic program

$$\begin{array}{ll} \min\limits_{\tilde{d}\in R^n} & \frac{1}{2}\langle (d_k + \tilde{d}), H_k(d_k + \tilde{d})\rangle + \langle \nabla f(x_k), d_k + \tilde{d}\rangle \\ \text{s.t.} & bl \leq x_k + d_k + \tilde{d} \leq bu \\ & g_j(x_k + d_k) + \langle \nabla g_j(x_k), \tilde{d}\rangle \leq -\min(\nu\|d_k\|, \|d_k\|^{2.5}), \quad j = 1,\ldots,n_i \\ & \langle c_j, x_k + d_k + \tilde{d}\rangle \leq d_j, \quad j = 1,\ldots,t_i - n_i \\ & \langle a_j, x_k + d_k + \tilde{d}\rangle = b_j, \quad j = 1,\ldots,l_e \end{array}$$

If the quadratic program has no solution or if $\|\tilde{d}_k\| > \|d_k\|$, set $\tilde{d}_k = 0$.

Step 2. Arc search. Compute t_k, the first number t in the sequence $\{1, \beta, \beta^2, \ldots\}$ satisfying

$$\begin{array}{l} f(x_k + td_k + t^2\tilde{d}_k) \leq f_k + \alpha t\langle \nabla f(x_k), d_k\rangle \\ g_j(x_k + td_k + t^2\tilde{d}_k) \leq 0, \quad j = 1,\ldots,n_i \end{array}$$

where f_k is equal to $f(x_k)$ for the Armijo line search and to $\max\limits_{l=0,\ldots,M} f(x_{k-l})$ for the nonmonotone line search. (The line search rule and, when appropriate, the value of M, are selected by the user: see explanation for `mode` in specification of FSQPD).

Step 3. Updates. Compute a new approximation H_{k+1} to the Hessian of the Lagrangian using the BFGS formula with Powell's modification[4]. Set $x_{k+1} = x_k + t_k d_k + t_k^2 \tilde{d}_k$. Increase k by one. Go back to Step 1.

□

If the initial guess x_0 provided by the user is not feasible, FSQP first solves a strictly convex quadratic program

$$\begin{array}{ll} \min\limits_{v \in R^n} & \langle v, v \rangle \\ s.t. & bl \leq x_0 + v \leq bu \\ & \langle c_j, x_0 + v \rangle \leq d_j, \quad j = 1, \ldots, t_i - n_i \\ & \langle a_j, x_0 + v \rangle = b_j, \quad j = 1, \ldots, l_e \end{array}$$

Then, starting from the feasible point in $R^{n+1} = (x_0 + v, \max\limits_{j=1,\ldots,n_i} g_j(x_0 + v))$, it will iterate, using the algorithm just stated, on the problem

$$\begin{array}{ll} \min\limits_{x \in R^n, \zeta \in R} & \zeta \\ \text{s.t.} & bl \leq x \leq bu \\ & g_j(x) \leq \zeta, \quad j = 1, \ldots, n_i \\ & \langle c_j, x \rangle \leq d_j, \quad j = 1, \ldots, t_i - n_i \\ & \langle a_j, x \rangle = b_j, \quad j = 1, \ldots, l_e \end{array}$$

until a value $\zeta \leq 0$ is reached. The corresponding iterate x will then be feasible for the original problem.

3 Specification of Subroutine FSQPD

Only a double precision version of FSQP, FSQPD is currently available. The specification of FSQPD is as follows:

```
      subroutine FSQPD(nparam,nnl,nineq,neq,mode,M,iprint,miter,inform,
     *                 bigbnd,eps,objef,bndl,bndu,x,g,iw,iwsize,w,nwsize,
     *                 obj,constr,gradob,gradcn)
      implicit double precision (a-h,o-z)
      dimension bndl(nparam),bndu(nparam),x(nparam),g(nineq+neq),
     *          iw(iwsize),w(nwsize)
      external obj,constr,gradob,gradcn
```

Important: all real variables and arrays must be declared as double precision in the routine that calls FSQPD. Followings are specifications of parameters and workspace.

`nparam` **(Input)** Number of free variables, i. e. , the dimension of `x`.

`nnl` **(Input)** Number (possibly zero) of nonlinear (inequality) constraints (n_i in the algorithm description).

`nineq` **(Input)** Total number (possibly equal to `nnl`) of inequality constraints (t_i in the algorithm description).

`neq` **(Input)** Number (possibly zero) of linear equality constraints (l_e in the algorithm description).

`mode` **(Input)** Defines the type of line search:

- `mode` = 0 : Armijo line search is selected, resulting in descent of the objective function at each iteration.
- `mode` = 1 : Nonmonotone line search is selected, resulting in descent of the objective function within `M` iterations, where `M` is given by the user. It is recommended to select `M` in the range $1 \leq$ `M` ≤ 10.

`M` **(Input)** Used only when `mode` = 1 (see above).

`iprint` **(Input)** Variable indicating the desired output:

- `iprint` = 0 : No information except for user-input errors is displayed.
- `iprint` = 1 : At the end of execution, status (`inform`), iterate, objective value, constraint values, number of evaluations of objective and nonlinear constraints, and norm of the Kuhn-Tucker vector are displayed.
- `iprint` = 2 : At the end of each iteration, the same information as with `iprint` = 1 is displayed.
- `iprint` = 3 : At each iteration, the same information as with `iprint` = 2, plus detailed information in the search direction computation (*Step* 1), in the line search (*Step* 2), and in the update (*Step* 3) is displayed.

`miter` **(Input)** Maximum number of iterations (*Steps* 1-3) allowed by the user before termination of execution.

`inform` **(Output)** Integer indicating the results of FSQPD.

`inform` = 0 :	Normal termination of execution in the sense that the norm of the final Kuhn-Tucker vector $\nabla L(x, \xi, \lambda, \mu)$ is no greater than `eps`.
`inform` = 1 :	The user-provided initial guess is infeasible and FSQPD is unable to generate a point satisfying all linear constraints.
`inform` = 2 :	The user-provided initial guess is infeasible and FSQPD is unable to generate a point satisfying all constraints.
`inform` = 3 :	The maximum number `miter` of iterations has been reached.
`inform` = 4 :	The line search fails to find a new iterate (step size being smaller than the machine precision `epsmac` (computed by FSQPD).
`inform` = 5 :	Failure in attempting to construct d^0.
`inform` = 6 :	Failure in attempting to construct d^1.
`inform` = 7 :	Input data are not consistent (with printout indicating the error).

`bigbnd` **(Input)** It plays the role of Infinite Bound.

`eps` **(Input)** Final norm requirement for the Kuhn-Tucker vector (ε in the algorithm description). It should be bigger than the machine precision `epsmac` (computed by FSQPD).

`objef` **(Output)** Value of objective function at the end of execution.

`bndl` **(Input)** Array of dimension `nparam` containing lower bounds for the components of `x`. To specify a non-existent lower bound (i. e. `bndl`$(j) = -\infty$ for some j), the value used must satisfy `bndl`$(j) \leq$ `-bigbnd`.

`bndu` **(Input)** Array of dimension `nparam` containing upper bounds for the components of `x`. To specify a non-existent upper bound (i. e. `bndu`$(j) = \infty$ for some j), the value used must satisfy `bndu`$(j) \geq$ `bigbnd`.

`x`	**(Input)** Initial guess. **(Output)** Final solution.
`g`	Array of dimension `nineq` + `neq`. **(Output)** Values of constraints at `x` at the end of execution.
`iw`	Integer workspace vector of dimension `iwsize`.
`iwsize`	**(Input)** Integer workspace length for `iw`. It must be at least as big as: $3 \times$ `nparam` $+ 2 \times \max(1,$ `nineq` + `neq`$) + 16$.
`w`	Double precision workspace of dimension `nwsize`.
`nwsize`	**(Input)** Workspace length for `w`. It must be at least as big as: $4 \times \mathtt{nparam}^2 + 2 \times \mathtt{nparam} \times \max(1, \mathtt{nineq}+\mathtt{neq}) + 22 \times \mathtt{nparam} + 10 \times \max(1, \mathtt{nineq}+\mathtt{neq}) + \mathtt{mm} + 21$. Here `mm` = `M` if `mode` = 1 and `mm` = 0 otherwise.
`obj`	**(Input)** Name of the user-defined function that computes the value of the objective function. This name must be declared as **external** in the calling routine and passed as an argument to FSQPD. The detailed specification is given in Section 5.1 below.
`constr`	**(Input)** Name of the user-defined function that computes the value of the constraints. This name must be declared as **external** in the calling routine and passed as an argument to FSQPD. The detailed specification is given in Section 5.2 below.
`gradob`	**(Input)** Name of the subroutine that computes the gradient of `obj`. This name must be declared as **external** in the calling routine and passed as an argument to FSQPD. The user must pass the subroutine name `grobfd` (and declare it as **external**), if he/she wishes that FSQPD evaluate this gradient automatically, by forward finite differences. The detailed specification is given in Section 5.3 below.
`gradcn`	**(Input)** Name of the subroutine that computes the gradients of the constraints. This name must be declared as **external** in the calling routine and passed as an argument to FSQPD. The user must pass the subroutine name `grcnfd` (and declare it as **external**), if he/she wishes that FSQPD evaluate these gradients automatically, by forward finite differences. The detailed specification is given in Section 5.4 below.

4 Description of the Output

No output will be displayed before a feasible starting point is obtained. The following information is displayed at the end of execution when `iprint` $= 1$ or at each iteration when `iprint` $= 2$:

`iteration`	Total number of iterations (`iprint` $= 1$) or iteration number (`iprint` $= 2$).
`inform`	See Section 3. It is displayed only at the end of execution.
`x`	Iterate.
`objective`	Value of objective function at `x`.
`maxobj`	(displayed only if `mode`$= 1$) Maximum value of objective function over the last `M` iterations (including the current one).
`constraints`	Values of the constraints at `x`.
`ncallf`	Number of times the objective function has been evaluated (so far).
`ncallc`	Number of evaluations of individual (scalar) nonlinear constraints (so far).
`ktnorm`	Norm of the Kuhn-Tucker vector at the current iteration. If execution terminates normally (`inform` $= 0$), then `ktnorm` $\leq$ `eps`.

For `iprint` $= 3$, in addition to the same information as the one for `iprint` $= 2$, the following is printed every iteration.

Details in the computation of a search direction (*Step* 1):

`d0norm`	Norm of the quasi-Newton direction d_k^0.
`d1norm`	Norm of the first order direction d_k^1.
`d`	Search direction $d_k = (1-\rho)d_k^0 + \rho d_k^1$.
`dnorm`	Norm of d_k.
`rho`	Coefficient ρ_k in $d_k = (1-\rho_k)d_k^0 + \rho_k d_k^1$.
`dtilde`	Correction $\tilde{d}_k$.

`dtnorm` Norm of the correction direction $\tilde{d}_k$.

Details in the line search (*Step* 2):

`trial step` The trial steplength t in the search direction.

`trial point` Trial iterate along the search arc with `trial step`.

`trial objective` Value of the objective function at `trial point`.

`trial constraints` Values of nonlinear constraints up to the first one which is not feasible at `trial point`.

Details in the updates (*Step* 3):

`gradf` Gradient of objective function at the new iterate.

`gradg` Gradients of constraints at the new iterate.

`multipliers` Multiplier estimates ordered as ξ's, λ's, and μ's (from quadratic program).

`hess` New estimate of the Hessian matrix of the Lagrangian.

5 User-Supplied Subroutines

At least two of the following four Fortran 77 subroutines, namely `obj` and `constr`, must be provided by the user in order to define the problem. The name of all four routines can be changed at the user's will, as they are passed as arguments to FSQPD.

5.1 Function obj

The function **obj**, to be provided by the user, computes the value of the objective function at a given iterate. The specification of **obj** for FSQPD is

```
      function obj(nparam,x)
      implicit double precision (a-h,o-z)
      dimension x(nparam)
c
c     assigns to obj the value of the objective function
c     evaluated at x
```

```
c
      return
      end
```

Arguments:

nparam	**(Input)** Dimension of x.
x	**(Input)** Current iterate.

5.2 Function constr

The function **constr**, to be provided by the user, computes the value of the constraints. The specification of `constr` for FSQPD is as follows

```
      function constr(nparam,j,x)
      implicit double precision (a-h,o-z)
      dimension x(nparam)
c
c     for given j, assigns to constr the value of the jth constraint
c     evaluated at x
c
      return
      end
```

Arguments:

nparam	**(Input)** Dimension of x.
j	**(Input)** Number of the constraint to be computed.
x	**(Input)** Current iterate.

The order of the constraints must be as follows. First the `nnl` (possibly zero) nonlinear inequality constraints. Then the `nineq-nnl` (possibly zero) linear inequality constraints. Finally, the `neq` (possibly zero) linear equality constraints.

5.3 Subroutine gradob

The subroutine **gradob** computes the gradient of the objective function. The specification of `gradob` for FSQPD is as follows

```
      subroutine gradob(nparam,rteps,x,objef,gradf,dummy)
      implicit double precision (a-h,o-z)
      dimension x(nparam),gradf(nparam)
c
c     assigns to gradf the gradient of the objective function
c     evaluated at x
c
      return
      end
```

The user may omit to provide this routine and require that forward finite difference approximation be used by FSQPD via calling `grobfd` instead (see argument `gradob` of FSQPD).

Arguments:

`nparam`	**(Input)** Dimension of `x`.
`rteps`	**(Input)** (used by `grobfd`) The square root of the machine precision `epsmac` (computed in FSQPD).
`x`	**(Input)** Current iterate.
`objef`	**(Input)** (used by `grobfd`) Value of the objective at `x`.
`dummy`	**(Input)** Used by `grobfd`.
`gradf`	**(Output)** Gradient of the objective function at x.

Note that `rteps`, `objef`, and `dummy` are passed as arguments to `gradob` to allow for forward finite difference computation of the gradient.

5.4 Subroutine gradcn

The subroutine **gradcn** computes the gradients of the constraints. The specification of `gradcn` for FSQPD is as follows

```
      subroutine gradcn (nparam,j,rteps,x,gj,gradg,dummy)
      implicit double precision (a-h,o-z)
      dimension x(nparam),gradg(nparam)
      external constr
c
c     assigns to gradg the gradient of the jth constraint
c     evaluated at x
c
      return
      end
```

The user may omit to provide this routine and require that forward finite difference approximation be used by FSQPD via calling `grcnfd` instead (see argument `gradcn` of FSQPD).

Arguments:

`nparam`	**(Input)** Dimension of `x`.
`j`	**(Input)** Number of constraint for which gradient is to be computed.
`rteps`	**(Input)** (used by `grcnfd`) The square root of the machine precision `epsmac` (computed by FSQPD).
`x`	**(Input)** Current iterate.
`gj`	**(Input)** (used by `grcnfd`) Value of the `j`th constraint at `x`.
`dummy`	**(Input)** Used by `grcnfd`.
`gradg`	**(Output)** Gradient of constraint `j` evaluated at `x`.

Note that `rteps`, `gj`, and `dummy` are passed as arguments to `gradcn` to allow for forward finite difference computation of the gradients.

6 Organization of FSQPD and Main Subroutines

6.1 Main Subroutines

FSQPD first checks for inconsistencies using the subroutine `check`. It then checks if the starting point given by the user satisfies the linear constraints and if not, generates a point satisfying these constraints using subroutine `initpt`. It then calls FSQPD1 for generating a point satisfying linear and nonlinear constraints. Finally, it attempts to find a point satisfying the optimality condition using again FSQPD1.

`check`	Checks that all upper bounds on variables are no smaller than lower bounds; checks that all input integers are nonnegative and appropriate (`nineq` $\geq$ `nnl`, etc.); and checks that `eps` is at least as large as the machine precision `epsmac` (computed by FSQPD).
`initpt`	Attempts to generate a feasible point satisfying simple bounds and all linear constraints.
`FSQPD1`	Main subroutine used twice by FSQPD, first for generating a feasible iterate as explained at the end of Section 2 and second for generating an optimal iterate from that feasible iterate.

FSQPD1 uses the following subroutines:

`dir`	Computes an arc search direction (*Step* 1 in algorithm FSQP).
`step`	Computes a step size along the arc (*Step* 2 in algorithm FSQP).
`hesian`	Performs the Hessian updates (*Step* 3 in algorithm FSQP).
`out`	Print the output for `iprint` = `1` or `iprint` = 2.
`grobfd`	(optional) Computes the gradient of the objective function by forward finite differences with mesh size equal to `rteps` $\times$ $\max(1.0, \lvert x_i \rvert)$ (`rteps` is the square root of `epsmac`, the machine precision computed by FSQPD).
`grcnfd`	(optional) Computes the gradient of a constraint by forward finite differences with mesh size equal to `rteps` $\times$ $\max(1.0, \lvert x_i \rvert)$ (`rteps` is the square root of `epsmac`, the machine precision computed by FSQPD).

6.2 Other Subroutines

In addition to QPSOL and subroutines associated with it, the following subroutines are used:

```
di0      di1      dtilde     error     ident     indexs
matrvc   nullvc   qphess     scaprd    small     subout
```

7 Example of Use

The following problem is borrowed from [5] (Problem 32). It contains simple bounds on the variables, nonlinear inequality constraints, linear equality constraints. The objective function f is defined for $x \in R^3$ by

$$f(x) = (x_1 + 3x_2 + x_3)^2 + 4(x_1 - x_2)^2$$

The constraints are

$$\begin{array}{ll} 0 \le x_i, & i = 1, \cdots, 3 \\ x_1^3 - 6x_2 - 4x_3 + 3 \le 0 & 1 - x_1 - x_2 - x_3 = 0 \end{array}$$

The feasible initial guess is: $x_0 = (0.1,\ 0.7,\ 0.2)^T$ with corresponding value of the objective function $f(x_0) = 7.2$. The final solution is: $x^* = (0.,\ 0.,\ 1.)^T$ with $f(x^*) = 1.$ A suitable main program is as follows.

```
c
c     problem description
c
      implicit double precision (a-h,o-z)
      dimension bndl(3),bndu(3),x(3),g(2),iw(29),w(275)
      external obj32,cntr32,grob32,grcn32
c
      nparam=3
      nnl=1
      nineq=1
      neq=1
c
      mode=0
      iprint=1
      miter=500
      bigbnd=1.d+10
      eps=1.d-08
      iwsize=29
      nwsize=275
c
      bndl(1)=0.
      bndl(2)=0.
      bndl(3)=0.
      bndu(1)=bigbnd
      bndu(2)=bigbnd
      bndu(3)=bigbnd
c
c     give the initial value of x
c
      x(1)=0.1
      x(2)=0.7
      x(3)=0.2
c
      call FSQPD(nparam,nnl,nineq,neq,mode,M,iprint,miter,inform,
     *           bigbnd,eps,objef,bndl,bndu,x,g,iw,iwsize,w,nwsize,
     *           obj32,cntr32,grob32,grcn32)
      stop
      end
```

Following are the subroutines defining the objective and constraints and their gradients.

```
      function obj32(nparam,x)
      implicit double precision (a-h,o-z)
      dimension x(nparam)
c
      obj32=(x(1)+3.d0*x(2)+x(3))**2+4.d0*(x(1)-x(2))**2
      return
      end
c
      function cntr32(nparam,j,x)
      implicit double precision (a-h,o-z)
      dimension x(nparam)
c
      go to (10,20),j
 10   cntr32=x(1)**3-6.d0*x(2)-4.d0*x(3)+3.d0
      return
 20   cntr32=1.d0-x(1)-x(2)-x(3)
      return
      end
c
      subroutine grob32(nparam,rteps,x,objef,gradf,dummy)
      implicit double precision (a-h,o-z)
      dimension x(nparam),gradf(nparam)
c
      fa=2.d0.*(x(1)+3.d0*x(2)+x(3))
      fb=8.d0*(x(1)-x(2))
      gradf(1)=fa+fb
      gradf(2)=3.d0*fa-fb
      gradf(3)=fa
      return
      end
c
      subroutine grcn32( nparam,j,rteps,x,gj,gradg,dummy)
      implicit double precision (a-h,o-z)
      dimension x(nparam),gradg(nparam)
c
      go to (10,20), j
 10   gradg(1)=3.d0*x(1)**2
      gradg(2)=-6.d0
      gradg(3)=-4.d0
      return
 20   gradg(1)=-1.d0
      gradg(2)=-1.d0
      gradg(3)=-1.d0
      return
      end
```

After running the algorithm on Sun 3/160, the following output is obtained:

```
 ------------- FSQPD   OUTPUT -------------
 The given initial point is feasible:
                     0.10000000000000e+00
                     0.70000000000000e+00
                     0.20000000000000e+00

 iteration            3
 inform               0
 x                   0.00000000000000e+00
                     0.00000000000000e+00
                     0.10000000000000e+01
```

```
objective           0.10000000000000e+01
constraints        -0.10000000000000e+01
                    0.00000000000000e+00
ktnorm              0.31401849173676e-15
ncallf               3
ncallc               6

Normal termination: You have obtained a solution !!
```

8 Results for Test Problems

Table 1 contains results obtained for some test problems from [5]. P.No. indicates the problem number as in [5], `nnl` the number of nonlinear constraints, `ncallf` the total number of evaluations of the objective function, `ncallc` the total number of evaluations of the (scalar) nonlinear constraint functions, `iter` the total number of iterations, `objective` the final value of the objective, `ktnorm` the norm of Kuhn-Tucker vector at the final iterate, `eps` the user-provided Kuhn-Tucker norm requirement. The value of parameter `mode` (the type of line search) is indicated in column "mode". `mode` = 1 with `M` = 5 was always attempted, wherever results are not provided they are identical to those obtained with `mode` = 0. All these results were obtained on SUN 3/160 with -ffpa compiling option.

Acknowledgment

The authors are indebted to Dr. E.R. Panier for a great many invaluable comments and suggestions.

P.No.	mode	nnl	ncallf	ncallc	iter	objective	ktnorm	eps
p12	0	1	7	19	7	−.300000000E+02	.43E−06	.10E-05
p29	0	1	13	22	10	−.226274170E+02	.72E−08	.10E-05
	1		12	22	11	−.226274170E+02	.11E−09	.10E-05
p30	0	1	16	31	16	.100000000E+01	.36E−08	.10E-07
p31	0	1	13	33	8	.600000000E+01	.17E−06	.10E-04
	1		9	31	9	.600000000E+01	.34E−05	.10E-04
p32	0	1	3	6	3	.100000000E+01	.31E−15	.10E-07
p33	0	2	4	14	4	−.400000000E+01	.12E−11	.10E-07
p34	0	2	7	28	7	−.834032445E+00	.54E−09	.10E-07
p43	0	3	15	88	10	−.440000000E+02	.84E−06	.10E-04
	1		10	71	10	−.440000000E+02	.84E−06	.10E-04
p44	0	0	6	0	6	−.150000000E+02	.0	.10E-07
p51	0	0	11	0	9	.505655658E−15	.68E−07	.10E-05
p57	0	1	15	17	3	.306463061E−01	.29E−05	.10E-04
p66	0	2	8	30	8	.518163274E+00	.47E−09	.10E-07
p76	0	0	6	0	6	.468181818E−01	.34E−04	.10E-03
p84	0	6	4	42	4	−.528033513E+07	.0	.10E-07
p86	0	0	8	0	7	−.323486790E+02	.12E−13	.10E-07
p93	0	2	14	66	11	.135076021E+03	.46E−03	.10E-02
	1		14	68	12	.135076021E+03	.18E−03	.10E-02
p100	0	4	37	237	16	.680630057E+03	.53E−04	.25E-03
	1		19	198	19	.680630057E+03	.21E−04	.25E-03
p110	0	0	12	0	9	−.457784697E+02	.76E−10	.10E-07
p113	0	5	12	150	12	.243063769E+02	.34E−03	.20E-02
p117	0	5	18	216	18	.323486790E+02	.33E−04	.10E-03
p118	0	0	19	0	19	.664820450E+03	.13E−14	.10E-07

Table 1: Results for Test Problems

References

[1] E.R. Panier & A.L. Tits, "On Feasibility, Descent and Superlinear Convergence in Inequality Constrained Optimization," Systems Research Center, University of Maryland, Technical Report SRC-TR-89-27, College Park, MD 20742, 1989.

[2] L. Grippo, F. Lampariello & S. Lucidi, "A Nonmonotone Line Search Technique for Newton's Method," *SIAM J. Numer. Anal.* 23 (1986).

[3] P.E. Gill, W. Murray, M.A. Saunders & M.H. Wright, "User's Guide for SOL/QPSOL: A FORTRAN Package for Quadratic Programming," Stanford Univ., Technical Report SOL 83-7, 1983.

[4] M.J.D. Powell, "A Fast Algorithm for Nonlinearly Constrained Optimization Calculations," in *Numerical Analysis, Dundee, 1977, Lecture Notes in Mathematics 630*, G.A. Watson, ed., Springer-Verlag, 1978, 144–157.

[5] W. Hock & K. Schittkowski, *Test Examples for Nonlinear Programming Codes*, Lecture Notes in Economics and Mathematical Systems (187), Springer Verlag, 1981.

SRC TR 89-63

Error Exponents for a Class of Multiterminal Detection Systems

By

H.M.H. Shalaby and A. Papamarcou

ERROR EXPONENTS FOR A CLASS OF MULTITERMINAL DETECTION SYSTEMS

Hossam M. H. Shalaby and Adrian Papamarcou

Systems Research Center and Electrical Engineering Department

University of Maryland, College Park, MD 20742

ABSTRACT

We discuss the asymptotic performance of a multiterminal detection system comprising a central decision maker and r remote sensors that have access to discrete, spatially dependent, and temporally memoryless observations. We assume that prior to transmitting information to the decision maker, each sensor compresses its data at a rate which approaches zero as the blocklength tends to infinity; and that on the basis of the compressed data from all sensors, the decision maker seeks to determine whether the data are governed by an r-variate distribution Q or by an alternative $\bar{Q}$. Under the classical criterion that stipulates minimization of the type II error rate subject to an upper bound ϵ on the type I error rate, we show that the error exponent achievable by such a system has a simple characterization, is independent of the value of ϵ, and is insensitive to changes in compression rate as long as the asymptotic rate on all sensors is zero.

ERROR EXPONENTS FOR A CLASS OF MULTITERMINAL DETECTION SYSTEMS

Hossam M. H. Shalaby and Adrian Papamarcou

Systems Research Center and Electrical Engineering Department

University of Maryland, College Park, MD 20742

1. Introduction

We consider the problem of testing a simple hypothesis H versus a simple alternative $\bar{H}$ on the basis of compressed data from a discrete-time, discrete-alphabet, memoryless multiple source. In its simplest form, our setup comprises a detector or decision maker linked to two remote sensors S_X and S_Y. The sensors S_X and S_Y observe the respective components of the random sequence $\{(X_i, Y_i)\}_{i=1}^{n}$, and encode their observations using a maximum of $nR_X(n)$ and $nR_Y(n)$ bits, respectively. Upon receipt of the two codewords, the detector accepts or rejects H in accordance with the classical criterion that stipulates minimization of the *type II error rate* (the probability of accepting H when $\bar{H}$ is true) subject to a fixed upper bound ϵ on the *type I error rate* (the probability of rejecting H when it is true).

Distributed detection systems of the above type have been widely studied in the recent literature. The models most frequently encountered employ compression rates $R_X(n)$ and $R_Y(n)$ such that the codeword lengths are constant in n and often equal to unity [7–10]. In such cases, the detector receives from each sensor what amounts to a local decision, possibly accompanied by an assessment (on a fixed finite-valued scale) of the sensor's confidence in that decision. Yet the information-theoretic formulation given above clearly allows more general forms for $R_X(n)$ and $R_Y(n)$; indeed, the models discussed in this paper and in [1–4] employ codebook sizes that are varying with n.

It is worth noting that one particular model, namely that for which $R_X(n) = R_Y(n) =$

∞, admits a rather simple analysis. In that case, neither sensor needs to compress its data, and the detector knows the observed sequence $\{(X_i, Y_i)\}_{i=1}^n$ precisely. The optimal decision rule is then specified by the Neyman-Pearson lemma, and the resulting minimum type II error rate $\beta_n(\epsilon)$ satisfies

$$-\lim_n \frac{1}{n} \log \beta_n(\epsilon) \;=\; D(P_{XY}||\bar{P}_{XY}) \;.$$

The quantity appearing on the left-hand side of the above equation (which is due to Stein [5]) is termed the *error exponent* for the hypothesis testing problem. On the right-hand side, $D(\cdot||\cdot)$ denotes informational divergence, and P_{XY} (resp. $\bar{P}_{XY}$) the distribution of (X_i, Y_i) under H (resp. $\bar{H}$).

Unfortunately, in the case of arbitrary $R_X(n)$ and $R_Y(n)$, the determination of the optimal detector is a highly complex task that also involves the optimization of the data encoders at S_X and S_Y. For this reason, it is preferable to study tractable compression/decision schemes which are *asymptotically* optimal, i.e., achieve the same error exponents as their optimal counterparts. The investigations in [2,3] are examples of such studies.

In [2], Ahlswede and Csiszár discussed the problem of one-sided fixed-rate compression (i.e., $R_X(n) = R_X$ and $R_Y(n) = \infty$). In the special case of hypothesis testing against independence (i.e., $\bar{P}_{XY} = P_X \times P_Y$), they obtained a single letter characterization of the error exponent by recourse to entropy characterization techniques. Also, in the general case where $\bar{P}_{XY} > 0$, they showed that the error exponent is independent of the value of the upper bound ϵ on the probability of type I error. Yet the problem of single-letter characterization of the error exponent in the case $\bar{P}_{XY} \neq P_X \times P_Y$ remained unsolved; single-letter lower bounds to that exponent were obtained in both [2] and [3] using compression/decision schemes whose asymptotic optimality was not established. In a somewhat different model involving exponentially decaying bounds on the type I error rate, Han and

Kobayashi [4] developed good upper bounds on the error exponent for two-sided fixed-rate compression.

In this paper we consider the problem of one-sided compression at asymptotically zero rate; in other words, we assume that $R_X(n) \to 0$ and $R_Y(n) = \infty$. Our inquiry was motivated by the study in [3] of one-sided one-bit compression, i.e., $nR_X(n) = 1$ and $R_Y(n) = \infty$. For that situation, Han proposed a simple scheme that compressed both S_X and S_Y to one bit, was invariant in ϵ, and yielded a simply characterized lower bound on the error exponent. He then established the tightness of the lower bound (and hence also the asymptotic optimality of the proposed scheme) for values of ϵ lying in an interval $(0, \epsilon_0)$, where $\epsilon_0 < 1$. In this work we extend the above results under the condition $\bar{P}_{XY} > 0$ by showing that

(a) the said lower bound is tight for all ϵ in $(0, 1)$;

(b) the proposed compression scheme, albeit employing one-bit compression of S_X and S_Y, is actually asymptotically optimal among all schemes such that $R_X(n) \to 0$ and $R_Y(n) = \infty$.

In other words, all systems effecting one-sided compression at asymptotically zero rate can be optimized so as to yield the same error exponent, which is independent of the actual rate constraint $R_X(n)$ and level ϵ for $\epsilon \in (0, 1)$.

The precise formulation of our problem is given in Section 2, and the main results appear in Section 3. In Section 4, we extend our results to hypothesis testing involving r-variate distributions, where $r > 2$.

2. Problem Statement and Preliminaries

(a) General notation. The observations of S_X and S_Y are denoted by the sequences $X^n = (X_1, \ldots, X_n) \in \mathcal{X}^n$ and $Y^n = (Y_1, \ldots, Y_n) \in \mathcal{Y}^n$, respectively, and the alphabets $\mathcal{X}$ and $\mathcal{Y}$ are assumed finite. Since the multiple source is memoryless, the sequence of

pairs $((X_1,Y_1),\ldots,(X_n,Y_n)) \in (\mathcal{X}\times\mathcal{Y})^n$ is i.i.d. under both hypotheses. In what follows, it will be convenient to deal with the product space $\mathcal{X}^n\times\mathcal{Y}^n$ instead of $(\mathcal{X}\times\mathcal{Y})^n$, and thus the observations will be collectively represented by the pair $(X^n,Y^n)\in\mathcal{X}^n\times\mathcal{Y}^n$.

By virtue of the aforementioned i.i.d. assumption, all distributions of interest can be specified through bivariate distributions on $\mathcal{X}\times\mathcal{Y}$. Under the null hypothesis, the distribution of any pair (X_i,Y_i) is denoted by P_{XY}, and its respective marginals by P_X and P_Y. The distributions of X^n, Y^n, and (X^n,Y^n) under the same hypothesis are denoted by P_X^n, P_Y^n and P_{XY}^n, respectively. The i.i.d. assumption then implies that for all (x^n,y^n) in $\mathcal{X}^n\times\mathcal{Y}^n$,

$$P_{XY}^n(x^n,y^n)=\prod_{i=1}^n P_{XY}(x_i,y_i) \ .$$

Analogous notation is employed for the alternative hypothesis, with $\bar{P}$ replacing P. We will also have occasion to use auxiliary distributions $\tilde{P}_{XY}$ and $\hat{P}_{XY}$ on $\mathcal{X}\times\mathcal{Y}$, which will yield marginals and higher-order distributions in the same manner as P_{XY} and $\bar{P}_{XY}$.

The compression of X^n is effected by the encoder f_n, where

$$f_n:\mathcal{X}\mapsto\{1,\ldots,M_n\} \ ,$$

and the codebook size M_n is constrained by

$$M_n\geq 2, \qquad \lim_n \frac{1}{n}\log M_n = 0 \ . \tag{2.1}$$

The corresponding detector is represented by the function

$$\phi_n:\{1,\ldots,M_n\}\times\mathcal{Y}^n\mapsto\{0,1\} \ ,$$

where the output 0 signifies the acceptance of the null hypothesis H, and 1 its rejection. This induces a partition of the original (i.e., non-compressed) sample space $\mathcal{X}^n\times\mathcal{Y}^n$ into an *acceptance* region

$$\mathcal{A}_n \overset{\text{def}}{=} \{(x^n,y^n)\in\mathcal{X}^n\times\mathcal{Y}^n : \phi_n(f_n(x^n),y^n)=0\} \ ,$$

and a *critical* (or rejection) region $\mathcal{D}_n = \mathcal{A}_n^c$. By nature of the encoding process, the acceptance region can be decomposed into M_n rectangles $C_i \times F_i$ in $\mathcal{X}^n \times \mathcal{Y}^n$ that possess disjoint projections C_i on $\mathcal{X}^n$. More precisely, if for every $1 \leq i \leq M_n$ we define

$$C_i = \{x^n \in \mathcal{X}^n : f_n(x^n) = i\} \qquad \text{and} \qquad F_i = \{y^n \in \mathcal{Y}^n : \phi_n(i, y^n) = 0\} \ ,$$

then we can write

$$\mathcal{A}_n = \bigcup_{i=1}^{M_n} C_i \times F_i \ , \qquad \text{where} \qquad (\forall i \neq j) \quad C_i \cap C_j = \emptyset \ . \tag{2.2}$$

(b) The hypothesis testing problem. In the above framework, the problem of testing $H : P$ versus $\bar{H} : \bar{P}$ can be formulated as follows: for a given level $\epsilon \in (0,1)$, minimize $\bar{P}^n_{XY}(\mathcal{A}_n)$ (the probability of type II error) over all acceptance regions $\mathcal{A}_n$ that

(C1) yield a value of $P^n_{XY}(\mathcal{A}_n^c)$ (probability of type I error) less than or equal to ϵ;

and

(C2) satisfy condition (2.2).

The resulting *minimum* probability of type II error is denoted by $\beta_n(M, \epsilon)$, and the associated error exponent is given by

$$\theta(M, \epsilon) \stackrel{\text{def}}{=} -\lim_n \frac{1}{n} \beta_n(M, \epsilon) \ ,$$

provided the limit on the right-hand side exists.

Note that in the absence of constraint (C2), the above formulation reduces to the ordinary hypothesis testing problem mentioned in the introduction, in which the optimal acceptance region is directly obtainable from the pointwise likelihood ratio

$$\bar{P}^n_{XY}(x^n, y^n)/P^n_{XY}(x^n, y^n)$$

(cf. the Neyman-Pearson lemma). The said constraint, however, denies us a compact representation of the optimal region, and we restrict ourselves to the study of the asymptotic behavior of $\beta_n(M, \epsilon)$ as n approaches infinity, a task which does not require precise knowledge of the optimal region.

More specifically, we show that under the positivity condition $\bar{P}_{XY} > 0$, for every sequence M_n satisfying (2.1) and every $\epsilon \in (0,1)$, the error exponent $\theta(M,\epsilon)$ is given by the minimum of the quantity

$$D(\tilde{P}_{XY} || \bar{P}_{XY})$$

over all bivariate distributions $\tilde{P}_{XY}$ on $\mathcal{X} \times \mathcal{Y}$ whose marginals on $\mathcal{X}$ and $\mathcal{Y}$ agree with those of P_{XY}. The positive result, namely the existence of a sequence of acceptance regions that achieve the above value, was shown in [3]. Our main result here is a strong converse, i.e., we show that for every value of $\epsilon \in (0,1)$ and every sequence of acceptance regions $\mathcal{A}_n$ satisfying constraints (C1) and (C2), the following is true:

$$-\liminf_n \frac{1}{n} \log \bar{P}^n_{XY}(\mathcal{A}_n) \leq \min_{\substack{\tilde{P}_{XY}: \\ \tilde{P}_X = P_X,\ \tilde{P}_Y = P_Y}} D(\tilde{P}_{XY} || \bar{P}_{XY}) .$$

(c) Typical sequences. Our proofs rely on the concept of a typical sequence, as developed in [6]. Here we cite some basic definitions and facts on typical sequences.

The *type* of a sequence $x^n \in \mathcal{X}^n$ is the distribution λ_x on $\mathcal{X}$ defined by the relationship

$$(\forall a \in \mathcal{X}) \qquad \lambda_x(a) \stackrel{\text{def}}{=} \frac{1}{n} N(a|x^n),$$

where $N(a|x^n)$ is the number of terms in x^n equal to a. The set of all types of sequences in $\mathcal{X}^n$, namely $\{\lambda_x : x^n \in \mathcal{X}^n\}$, will be denoted by $\mathcal{P}_n(\mathcal{X})$.

Given a type $\hat{P}_X \in \mathcal{P}_n(\mathcal{X})$, we will denote by $\hat{T}^n_X$ the set of sequences $x^n \in \mathcal{X}^n$ of type $\hat{P}_X$:

$$\hat{T}^n_X \stackrel{\text{def}}{=} \{x^n \in \mathcal{X}^n : \lambda_x = \hat{P}_X\} .$$

Also, for an arbitrary distribution $\tilde{P}_X$ on $\mathcal{X}$ and a constant $\eta > 0$, we will denote by $\tilde{T}^n_{X,\eta}$ the set of $(\tilde{P}_X, \eta)$-*typical* sequences in $\mathcal{X}^n$. A sequence x^n is $(\tilde{P}_X, \eta)$-typical if $|\lambda_x(a) - \tilde{P}_X(a)| \leq \eta$ for every letter $a \in \mathcal{X}$ and, in addition, $\lambda_x(a) = 0$ for every a such that $\tilde{P}_X(a) = 0$. Thus, if $||\cdot||$ denotes the sup norm and $\ll$ denotes absolute continuity, we have

$$\tilde{T}^n_{X,\eta} \stackrel{\text{def}}{=} \{x^n \in \mathcal{X}^n \colon ||\lambda_x - \tilde{P}_X|| \leq \eta,\ \lambda_x \ll \tilde{P}_X\} .$$

In the same manner, we will denote by $T^n_{X,\eta}$ the set of (P_X, η)- typical sequences in $\mathcal{X}^n$.

The proofs of the following lemmas appear in [6]. As usual, $|\mathcal{A}|$ denotes the cardinality of $\mathcal{A}$.

LEMMA 2.1. *For any $\hat{P}_X$ in $\mathcal{P}_n(\mathcal{X})$,*

$$(n+1)^{-|\mathcal{X}|} \exp[nH(\hat{P}_X)] \leq |\hat{T}^n_X| \leq \exp[nH(\hat{P}_X)] ,$$

and

$$(n+1)^{-|\mathcal{X}|} \exp[-nD(\hat{P}_X || \bar{P}_X)] \leq \bar{P}^n_X(\hat{T}^n_X) \leq \exp[-nD(\hat{P}_X || \bar{P}_X)] .$$

LEMMA 2.2. *For any distribution P_X on $\mathcal{X}$ and $\eta > 0$,*

$$P^n_X(T^n_{X,\eta}) \geq 1 - \frac{|\mathcal{X}|}{4n\eta^2} .$$

One can easily modify the above exposition to accommodate pairs $(x^n, y^n) \in \mathcal{X}^n \times \mathcal{Y}^n$ by reverting to their representation in $(\mathcal{X} \times \mathcal{Y})^n$. Thus the type of (x^n, y^n) is the distribution λ_{xy} on $\mathcal{X} \times \mathcal{Y}$ such that

$$\lambda_{xy}(a, b) = \frac{1}{n} \Big| \{ i : (x_i, y_i) = (a, b) \} \Big| ,$$

and the sets $\mathcal{P}_n(\mathcal{X} \times \mathcal{Y})$, as well as $\hat{T}^n_{XY} \subset \mathcal{X}^n \times \mathcal{Y}^n$ and $\tilde{T}_{XY,\eta} \subset \mathcal{X}^n \times \mathcal{Y}^n$, are defined accordingly.

In the following section we will omit the superscript n from T, as n will be essentially constant.

3. The main theorems.

THEOREM 3.1. *Let P_{XY} be arbitrary, and $\bar{P}_{XY} > 0$. For all $\epsilon \in (0,1)$ and sequences M_n and $\mathcal{A}_n$ satisfying conditions (2.1) and (2.2) respectively, the following is true: if for every n,*

$$P^n_{XY}(\mathcal{A}^c_n) \leq \epsilon \ ,$$

then

$$-\liminf_n \frac{1}{n} \log \bar{P}^n_{XY}(\mathcal{A}_n) \leq \min_{\substack{\tilde{P}_{XY}: \\ \tilde{P}_X = P_X,\ \tilde{P}_Y = P_Y}} D(\tilde{P}_{XY} \| \bar{P}_{XY}) \ .$$

PROOF. By (2.2), we have

$$\mathcal{A}_n = \bigcup_{i=1}^{M_n} C_i \times F_i \ ,$$

where the C_i's are pairwise disjoint. Assume that $P^n_{XY}(\mathcal{A}^c) \leq \epsilon$, or equivalently, $P^n_{XY}(\mathcal{A}) \geq 1-\epsilon$. Then there exists an index i_0 such that

$$P^n_{XY}(C_{i_0} \times F_{i_0}) \geq \frac{1-\epsilon}{M_n} \ .$$

Letting $C = C_{i_0}$ and $F = F_{i_0}$, we can rewrite the above as

$$P^n_{XY}(C \times F) \geq \exp(-n\delta_n), \tag{3.1}$$

where

$$\delta_n = -\frac{1}{n} \log(1-\epsilon) + \frac{1}{n} \log M_n \ ,$$

and $\delta_n \to 0$ by condition (2.1). Equation (3.1) clearly implies that

$$P^n_X(C) \geq \exp(-n\delta_n) \qquad \text{and} \qquad P^n_Y(F) \geq \exp(-n\delta_n) \ . \tag{3.2}$$

Thus asymptotically, neither C nor F has "exponentially small" probability. By the blowing-up lemma [6], this fact implies that both sets possess Hamming k_n-neighborhoods which are asymptotically "as thin" as the sets themselves (i.e., $k_n/n \to 0$), and whose probabilities approach unity as n tends to infinity. More precisely, let $d(\cdot,\cdot)$ denote Hamming distance, and define the Hamming k-neighborhood $\Gamma^k C$ of C by

$$\Gamma^k C \stackrel{\text{def}}{=} \{u^n \in \mathcal{X}^n \ : \ (\exists\, x^n \in C)\ d(x^n, u^n) \le k\} \ .$$

The blowing-up lemma asserts that under condition (3.2), there exist sequences k_n and γ_n satisfying

$$k_n/n \to 0 \qquad \text{and} \qquad \gamma_n \to 0 \ ,$$

and such that

$$P_X^n(\Gamma^{k_n} C) \ \ge\ 1-\gamma_n \qquad \text{and} \qquad P_Y^n(\Gamma^{k_n} F) \ \ge\ 1-\gamma_n \ . \tag{3.3}$$

In what follows, we will use k instead of k_n in all superscripts.

Equation (3.3) clearly holds true if we replace P by $\tilde{P}$, where $\tilde{P}_{XY}$ satisfies the marginal constraints

$$\tilde{P}_X = P_X \qquad \text{and} \qquad \tilde{P}_Y = P_Y \ .$$

Using the elementary property $\Pr(A \cap B) \ge \Pr(A) + \Pr(B) - 1$, we then obtain

$$\tilde{P}_{XY}^n(\Gamma^k C \times \Gamma^k F) \ \ge\ \tilde{P}_X^n(\Gamma^k C) \ +\ \tilde{P}_Y^n(\Gamma^k F) \ -\ 1 \ ,$$

and hence

$$\tilde{P}_{XY}^n(\Gamma^k C \times \Gamma^k F) \ \ge\ 1 - 2\gamma_n \ . \tag{3.4}$$

Thus under the n-fold product of $\tilde{P}_{XY}$, the probability of the rectangle $\Gamma^k C \times \Gamma^k F$ approaches unity as n tends to infinity. By Lemma 2.2, the same is true of the set of $\tilde{T}_{XY,\eta}$-typical elements in $\mathcal{X}^n \times \mathcal{Y}^n$, since

$$\tilde{P}_{XY}^n(\tilde{T}_{XY,\eta}) \ \ge\ 1 - \frac{|\mathcal{X}||\mathcal{Y}|}{4n\eta^2} \ .$$

Hence, for all sufficiently large n, we obtain

$$\tilde{P}^n_{XY}((\Gamma^k C \times \Gamma^k F) \cap \tilde{T}_{XY,\eta}) \geq \frac{1}{2} . \tag{3.5}$$

By definition of $\tilde{T}_{XY,\eta}$, we have the following decomposition:

$$\tilde{T}_{XY,\eta} = \bigcup_{\substack{\hat{P}_{XY} \in \mathcal{P}_n(\mathcal{X} \times \mathcal{Y}): \\ ||\hat{P}_{XY} - \tilde{P}_{XY}|| \leq \eta, \\ \hat{P}_{XY} \ll \tilde{P}_{XY}}} \hat{T}_{XY} .$$

Thus, observing that the elements of a given $\hat{T}_{XY}$ are equiprobable under any i.i.d. measure, we can rewrite (3.5) as

$$\sum_{\substack{\hat{P}_{XY} \in \mathcal{P}_n(\mathcal{X} \times \mathcal{Y}): \\ ||\hat{P}_{XY} - \tilde{P}_{XY}|| \leq \eta, \\ \hat{P}_{XY} \ll \tilde{P}_{XY}}} \tilde{P}^n_{XY}(\hat{T}_{XY}) \frac{|(\Gamma^k C \times \Gamma^k F) \cap \hat{T}_{XY}|}{|\hat{T}_{XY}|} \geq \frac{1}{2} .$$

At least one of the fractions in the above sum must be greater than or equal to 1/2; hence there exists a type $\hat{P}_{XY} \in \mathcal{P}_n(\mathcal{X} \times \mathcal{Y})$ satisfying

$$||\hat{P}_{XY} - \tilde{P}_{XY}|| \leq \eta \qquad \text{and} \qquad \hat{P}_{XY} \ll \tilde{P}_{XY} ,$$

and such that

$$\frac{|(\Gamma^k C \times \Gamma^k F) \cap \hat{T}_{XY}|}{|\hat{T}_{XY}|} \geq \frac{1}{2} .$$

Since pairs (x^n, y^n) of the same type are also equiprobable under $\bar{P}^n_{XY}$, we conclude that for the above type $\hat{P}_{XY}$,

$$\begin{aligned} \bar{P}^n_{XY}(\Gamma^k C \times \Gamma^k F) &\geq \bar{P}^n_{XY}((\Gamma^k C \times \Gamma^k F) \cap \hat{T}_{XY}) \\ &= \bar{P}^n_{XY}(\hat{T}_{XY}) \frac{|(\Gamma^k C \times \Gamma^k F) \cap \hat{T}_{XY}|}{|\hat{T}_{XY}|} \geq \frac{1}{2} \bar{P}^n_{XY}(\hat{T}_{XY}) . \end{aligned} \tag{3.6}$$

We have thus established that the probabilities of the sets $\Gamma^k C \times \Gamma^k F$ and $\hat{T}_{XY}$ are of the same exponential order under $\bar{P}^n_{XY}$. We now show that the same is true of the pair $\Gamma^k C \times \Gamma^k F$ and $C \times F$.

Consider an arbitrary element (u^n, v^n) of $\Gamma^k C \times \Gamma^k F$. By definition of Γ^k, there exists at least one element $(x^n, y^n) \in C \times F$ such that (x_i, y_i) differs from (u_i, v_i) for at most $2k_n$ values of i. We thus have

$$\begin{aligned} \bar{P}^n_{XY}(u^n, v^n) &= \prod_{i=1}^{n} \bar{P}_{XY}(u_i, v_i) \\ &\leq \rho^{-2k} \prod_{i=1}^{n} \bar{P}_{XY}(x_i, y_i) = \rho^{-2k} \bar{P}^n_{XY}(x^n, y^n) , \end{aligned} \tag{3.7}$$

where

$$\rho \overset{\text{def}}{=} \min_{x \in \mathcal{X},\ y \in \mathcal{Y}} \bar{P}_{XY}(x, y) > 0 .$$

As (u^n, v^n) ranges over $\Gamma^k C \times \Gamma^k F$, each element (x^n, y^n) of $C \times F$ will be selected at most $|\Gamma^k(x^n)| \cdot |\Gamma^k(y^n)|$ times. By virtue of this, (3.7) yields

$$\bar{P}^n_{XY}(\Gamma^k C \times \Gamma^k F) \leq \rho^{-2k} |\Gamma^k(x^n)| |\Gamma^k(y^n)| \bar{P}^n_{XY}(C \times F) .$$

From [6] we have the upper bound

$$|\Gamma^k(x^n)| \leq \exp\left[n\left(h\left(\frac{k_n}{n}\right) + \frac{k_n}{n} \log |\mathcal{X}|\right)\right] ,$$

where $h(\cdot)$ denotes the binary entropy function. Thus we may write

$$\bar{P}^n_{XY}(\Gamma^k C \times \Gamma^k F) \leq \exp(n\xi_n) \bar{P}^n_{XY}(C \times F) , \tag{3.8}$$

where

$$\xi_n = 2h\left(\frac{k_n}{n}\right) + \frac{k_n}{n} \log(|\mathcal{X}||\mathcal{Y}|) - \frac{2k_n}{n} \log \rho \to 0 .$$

As a final step, we combine equations (3.6) and (3.8) with the upper bound on $\bar{P}^n_{XY}(\hat{T}_{XY})$ provided by Lemma (2.1). Thus

$$\begin{aligned} \bar{P}^n_{XY}(C \times F) &\geq \frac{1}{2} \exp(-n\xi_n) \bar{P}^n_{XY}(\hat{T}_{XY}) \\ &\geq \frac{(n+1)^{-|\mathcal{X}|}}{2} \exp[-n(D(\hat{P}_{XY} || \bar{P}_{XY}) + \xi_n)] \\ &\geq \exp[-n(D(\hat{P}_{XY} || \bar{P}_{XY}) + \nu)], \end{aligned}$$

for every $\nu > 0$ and n sufficiently large. By continuity of the divergence functional, we can choose the typicality constant η such that

$$||\hat{P}_{XY} - \bar{P}_{XY}|| \leq \eta \qquad \Rightarrow \qquad \left|D(\hat{P}_{XY}||\bar{P}_{XY}) - D(\tilde{P}_{XY}||\bar{P}_{XY})\right| \leq \nu \ ,$$

whence we obtain that

$$\bar{P}^n_{XY}(C \times F) \ \geq \ \exp[-n(D(\tilde{P}_{XY}||\bar{P}_{XY}) + 2\nu)] \ ,$$

and consequently

$$-\liminf_n \frac{1}{n} \log \bar{P}^n_{XY}(\mathcal{A}_n) \ \leq \ D(\tilde{P}_{XY}||\bar{P}_{XY}) + 2\nu \ .$$

Since ν is arbitrary and $\tilde{P}_{XY}$ satisfies the appropriate marginal constraints, the proof is complete. $\triangle$

The above result, in conjunction with the positive part of Theorem 5 in [3], yields

THEOREM 3.2. *If $\bar{P}_{XY} > 0$, the error exponent for the hypothesis testing problem formulated in Section 2 is given by*

$$\theta(M, \epsilon) \ = \ \min_{\substack{\tilde{P}_{XY}: \\ \tilde{P}_X = P_X, \ \tilde{P}_Y = P_Y}} D(\tilde{P}_{XY}||\bar{P}_{XY})$$

for all $\epsilon \in (0, 1)$ and sequences M_n satisfying condition (2.1). $\triangle$

4. Extensions and concluding remarks

In the case of a discrete memoryless multiple source with r components, where $r > 2$, the salient problem becomes one of testing $H : Q$ versus $H : \bar{Q}$, where Q and $\bar{Q}$ are r-variate distributions, and $\bar{Q} > 0$. Again, one assumes that $\bar{Q} > 0$, and that all source components but one are compressed at asymptotically zero rate. One can then prove an analog of Theorem 3.2 stating that the error exponent is given by the minimum of $D(\tilde{Q}||\bar{Q})$ over all r-variate distributions $\tilde{Q}$ whose univariate marginals agree with those of Q. We give a sketch of the proof for the case of three sensors S_X, S_Y and S_Z, the first two of which are compressed at asymptotically zero rate. Here $Q = P_{XYZ}$ and $\bar{Q} = \bar{P}_{XYZ}$.

Direct part. Following the proof of Theorem 5 [3], we propose a sequence of acceptance regions $\mathcal{A}_n$ in $\mathcal{X}^n \times \mathcal{Y}^n \times \mathcal{Z}^n$ defined by

$$\mathcal{A}_n = T^n_{X,\eta} \times T^n_{Y,\eta} \times T^n_{Z,\eta} ,$$

where η is chosen so that the sup norm distance between the univariate marginals of P_{XYZ} and $\bar{P}_{XYZ}$ is at least 2η. One can easily show that $\mathcal{A}_n \supset T^n_{XYZ,\zeta}$ for suitable $\zeta > 0$, and thus by Lemma 2.2.,

$$P^n_{XYZ}(\mathcal{A}_n) \geq 1 - \epsilon$$

for all sufficiently large n. Also, by the type-counting argument given in [3], one can establish the relationship

$$-\lim_n \frac{1}{n} \log \bar{P}^n_{XYZ}(\mathcal{A}_n) = \min_{\substack{\tilde{P}_{XYZ}: \\ \tilde{P}_X = P_X,\ \tilde{P}_Y = P_Y, \\ \tilde{P}_Z = P_Z}} D(\tilde{P}_{XYZ} \| \bar{P}_{XYZ}) .$$

Converse part. Assuming that X^n and Y^n are compressed to a maximum of M_n and N_n bits, respectively, we can write any acceptance region $\mathcal{A}_n$ in $\mathcal{X}^n \times \mathcal{Y}^n \times \mathcal{Z}^n$ as

$$\mathcal{A}_n = \bigcup_{i=1}^{M_n} \bigcup_{j=1}^{N_n} C_i \times E_j \times F_{ij} .$$

Here the F_{ij}'s are subsets of $\mathcal{Z}^n$, and the C_i's and E_j's form partitions of $\mathcal{X}^n$ and $\mathcal{Y}^n$, repectively. Thus there exists a subset $C \times E \times F$ of $\mathcal{A}_n$ such that

$$P^n_{XYZ}(C \times E \times F) \geq \frac{1-\epsilon}{M_n N_n} .$$

Since the asymptotic zero rate condition (2.1) is also valid for N_n, one obtains the counterpart of (3.1), namely

$$P^n_{XYZ}(C \times E \times F) \geq \exp(-n\delta_n) ,$$

and the proof proceeds as before. $\triangle$

Note that the proof of the positive theorem involved one-bit compression per source component. Thus under any criterion based solely on the error exponent, distributed detection systems employing one-bit compression per sensor can attain the same asymptotic performance as more complex systems employing asymptotically zero-rate compression on $r-1$ sensors, and no compression at all on the remaining sensor.

The above result also covers the general case of a multiple source with $r = r_c + r_u$ components, r_c of which are compressed at asymptotically zero rate, while the remaining r_u are not compressed. The error exponent in this situation is given by the minimum of $D(\tilde{Q}||\bar{Q})$ over all distributions $\tilde{Q}$ that agree with Q on

(i) the univariate marginals corresponding to the compressed source components; and

(ii) the r_u-variate marginal corresponding to the r_u components which are not compressed.

Thus the latter r_u components are essentially treated as one. It also follows easily that in the case $r_u > 1$, the error exponent attainable by systems of the above type is *higher* than the corresponding figure for systems with asymptotically zero-rate compression on all components (unlike the case $r_u = 1$ discussed above).

The positivity assumption on the alternative hypothesis is pivotal to the proof of the converse result (Theorem 3.1); without this assumption, the application of the blowing-up lemma fails. The same difficulty was encountered in the proof of the converse result in [2, Theorem 6], which also employed the blowing-up lemma. We hope that this obstacle will eventually be removed.

In the meantime, we note that we have found instances where $\bar{Q} \not> 0$, and $D(\tilde{Q}||\bar{Q})$ is trivially minimized by $\tilde{Q} = Q$. In such cases, the resulting minimum is equal to the error exponent in the case of no data compression (cf. Stein's lemma), and the converse result follows immediately. We should also note that Theorem 3.2 does not subsume its counterpart in [3]. Although the converse theorem appearing in that work was valid for one-bit compression of S_X and for $\epsilon \in (0, \epsilon_0)$ only, the hypothesis of that theorem did not impose any constraints on $\bar{P}_{XY}$ other than $D(P_{XY}||\bar{P}_{XY}) < \infty$.

REFERENCES

[1] T. Berger, "Decentralized estimation and decision theory," presented at the IEEE Seven Springs Workshop on Information Theory, Mt. Kisco, NY, September 1979.

[2] R. Ahlswede and I. Csiszár, "Hypothesis testing with communication constraints," *IEEE Trans. Inform. Theory*, vol. IT-32, pp. 533-542, July 1986.

[3] T. S. Han, "Hypothesis testing with muliterminal data compression," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 759-772, Nov. 1987.

[4] T. S. Han and K. Kobayashi, "Exponential-Type Error Probabilities for Multiterminal Hypothesis Testing," *IEEE Trans. Inform. Theory*, vol. IT-35, pp. 2-14, Jan. 1989.

[5] S. Kullback, *Information Theory and Statistics.* New York: Wiley, 1959.

[6] I. Csiszár and J. Körner, *Information Theory: Coding Theorems for Discrete Memoryless Systems.* New York: Academic, 1982 and Budapest, Hungary: Akadémiai Kiadó, 1981.

[7] R. R. Tenney and N. R. Sandell, "Detection with distributed sensors," *IEEE Trans. Aerospace and Electronic Sys.*, vol. AES-17, pp. 501-510, July 1981.

[8] J. J. Chao and C. C. Lee, "A distributed detection scheme based on soft local decisions," *the 24th Annual Allerton Conference on Communication, Control, and Computing, Monticello, Illinois,* Oct. 1-3, 1986.

[9] S. C. Thomopoulos, R. Viswanathan, and D. C. Bougoulias, "Optimal decision fusion in multiple sensor systems," *IEEE Trans. Aerospace and Electronic Sys.*, vol. AES-23, pp. 644-653, Sep. 1987.

[10] A. R. Reibman and L. W. Nolte, "Optimal detection and performance of distributed sensor systems," *IEEE Trans. Aerospace and Electronic Sys.*, vol. AES-23, pp. 24-30, Jan. 1987.