2021 60th IEEE Conference on Decision and Control (CDC)
December 13-15, 2021. Austin, Texas

Distributed Computation of a Robust Estimator Based on Cauchy Noises

Nathaniel Snyder†, Moshe Idan‡, and Jason L. Speyer†

2021 60th IEEE Conference on Decision and Control (CDC) | 978-1-6654-3659-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/CDC45484.2021.9682987

***Abstract*—A real-time, recursive, multivariate estimation algorithm for time-invariant and time-varying linear systems with modelled Cauchy noises is developed. When previously compared to the Kalman Filter, the Multivariate Cauchy Estimator was shown to be robust against impulsive disturbances in the process or measurement functions, but proved computationally intractable for real-time estimation applications. Two significant insights allow for a reformulation of the Multivariate Cauchy Estimator to possess a streamlined recursive and computationally reduced characteristic function of the conditional probability density function of the system state-vector given the measurement sequence. This characteristic function is represented by a sum of terms, expanding with each measurement. First, we show that a cell-enumeration matrix can be computed for each hyperplane arrangement embedded within each term of the characteristic function of the Cauchy Estimator. We then show that functions used to formulate the terms of this characteristic function can be expressed as a vector of parameters operating on basis functions constructed from this enumeration matrix. This vector is obtained by solving an underdetermined system of equations. We demonstrate that our reformulation allows all terms with equal hyperplane arrangements to be reduced into a unique set. Secondly, we take advantage of advances in parallel processing to exploit the inherent parallelism found in the characteristic function of the Cauchy Estimator. A three state time-invariant system example is used to illustrate the performance of the Cauchy Estimator against the Kalman Filter when subjected to Gaussian and Cauchy noises. We report computational savings of over 99% when compared to the previous formulation. Furthermore, we discuss the real-time architecture of the Cauchy Estimator and report the execution speeds for a three-state system implemented on a single NVIDIA GeForce GTX 1060 graphics processing unit (GPU).**

***Index Terms*—control-systems, stochastic estimation, bayesian estimation, CUDA-C programming, GPU linear programming, cell enumeration, Cauchy pdfs**

I. Introduction

State estimation schemes traditionally assume Gauss-Markov models of the system's underlying process and measurement noises, leading to computationally efficient variants of the Kalman Filter [1, 2]. In many applications, volatile and impulsive fluctuations in the process or measurement functions can occur, which can be better described (probabilistically) by heavier tailed distributions [3, 4]. Such examples include replacing the Gaussian probability density function (pdf) in a Kalman Filter with heavy tailed Students-t distributions [5].

In [4], the Multivariate Cauchy Estimator was shown to analytically contain such robustness features against volatile process and measurement noises. Although no physical process is explicitly Cauchy distributed, since its tails over bound other realistic densities, estimators that are based on the Cauchy pdfs are hypothesized to be robust to unknown physical densities. The Cauchy Estimator demonstrated superior performance to that of the Kalman Filter in the presence of heavy-tailed Cauchy noise, and behaved similarly to the Kalman Filter in the presence of Gaussian noise [6]. However, in the previous formulation of the Cauchy Estimator, two major computational challenges existed when computing the conditional mean and variance from the characteristic function of the conditional pdf of the system state-vector given the measurement sequence, that is the central entity of the Cauchy Estimator. First, since both the conditional mean and variance are functions of the parameters of the current and past characteristic functions, the entire history of those parameters had to be stored. Second, at each estimation step, the number of parameters which make up the characteristic function grows factorially, without the ability to reduce the number of terms in the estimator. This was seen to make the computational burden of the estimator and its storage requirements intractable after only several estimation steps.

Motivated by the aforementioned, a reformulation of the Cauchy Estimator is developed in this paper to eliminate the need to store past parameters and moreover to reduce the number of terms that make up the characteristic function at each step. At its core, the characteristic function is represented by many *terms*, where each *term* i contains a central arrangement of m-hyperplanes in dimension d, denoted A_i. These terms involve a complex valued function that is constant in each cell of this hyperplane arrangement. The key component in the reformulated Cauchy Estimator for reducing the computational burden and memory of the estimator is a GPU driven incremental enumeration algorithm based on [7], to enumerate all the cells of hyperplane arrangements embedded within the terms of the characteristic function.

The central idea of the reformulated Cauchy Estimator is to quickly build an 'enumeration' matrix B_i for the hyperplane arrangements of each term $i \in [1, 2, \ldots, N_t]$, where N_t is the number of terms in the characteristic function. The matrix B_i is the result of running an efficient cell enumeration algorithm for enumeration of centralized hyperplane arrangements A_i. Each row of the enumeration matrix B_i holds a sequence of ± 1 sign values (sign-vector) that indicate in which halfspace a point in space lies with respect to each m_i hyperplanes in the arrangement. We refer the reader to [8] for a detailed introduction to cell enumeration.

The enumeration matrix B_i serves two purposes. Initially, it is used to compute the complex values held constant in each cell of the associated term i, of the related hyperplane arrangement. More significantly, B_i can be used to parameterize the values held constant in each cell through a linear system of equations, detailed in Sections III-B and V. Parameterizing these constant cell values is crucial in our reformulation, and is necessary to compute the conditional moments. Furthermore, as shown in Section V, all terms with identical hyperplane arrangements embedded within the Cauchy Estimator's characteristic function can be combined, thus drastically reducing the number of terms and hence the computation burden of the estimator. This reduction was not possible in the original formulation of the estimator in [4]. Hints of this structure were given in [6] for the two-dimensional system, but were not established in general.

The main contribution of this paper is to reformulate the characteristic function of the Multivariate Cauchy Estimator presented in [4] into a computationally reduced structure that is applicable

This work was supported in part by the National Science Foundation under grant numbers NSF/ENG/ECCS-BSF 1607502 and 1934467 as well as the NSF-BSF ECCS under Grant No. 2019639.

† N. Snyder and J. L. Speyer are with the Department of Mechanical and Aerospace Engineering, UCLA. Email: natsnyder1@g.ucla.edu, speyer@g.ucla.edu

‡ M. Idan is with the Faculty of Aerospace Engineering, Technion, Haifa, Israel. Email: moshe.idan@technion.ac.il.

for real-time multivariate estimation problems. The estimator in [4] is briefly reviewed in Section II. In Section III, we discuss a new alternative representation of the terms of the characteristic function that uses the notions of related hyperplane arrangements. In Section IV we describe the 'Incremental-Enumeration' procedure adopted from [7] to form the enumeration matrix B_i for the hyperplane arrangement A_i of each term of this characteristic function. Also, a minor, but crucial change for parallelizing the Incremental-Enumeration procedure allows our CUDA-C implementation of the Multivariate Cauchy Estimator to achieve real-time rates. The 'flattening' algorithm is discussed in Section V, which eliminates the recursive structure of the characteristic function in [4], commenting also on an algorithm for combining terms in the characteristic function. Section VI validates that the reformulated estimator computes the numerical linear time-invariant simulation results presented in [4], but does so with a vastly reduced characteristic function and at real-time rates. Additionally, we remark on how the real-time rates for time-varying three state systems would be similar to the time-invariant example provided. Concluding remarks are offered in section VII.

II. The Multivariate Cauchy Estimator

As detailed in [4], the Cauchy Estimator uses a characteristic function of the unnormalized conditional probability density function (ucpdf) of the system state-vector given the measurement history to generate the conditional mean and variance estimates of the state. This paper omits the derivation of the Cauchy Estimator and refers the reader to [4]. To make the presentation less mathematically cumbersome, we provide a loose parsing of the rigorous original notation in [4] and give a general overview of the structure of this characteristic function. The parameters and equations derived in [4] that are relevant to the reformulated estimator are re-presented in this section.

Given a discrete-time linear system of equations

$$x_{k+1} = \Phi x_k + \Gamma w_k, \tag{1a}$$
$$z_k = H x_k + v_k, \tag{1b}$$

with state $x_k \in \mathbb{R}^d$, dynamics matrix $\Phi \in \mathbb{R}^{d\times d}$, process noise input matrix $\Gamma \in \mathbb{R}^{d\times r}$, and process noise $w_k \in \mathbb{R}^r$. We let z_k model the measurement, with measurement matrix $H \in \mathbb{R}^{1\times d}$ and additive measurement noise w_k. Note that Φ, Γ, and H can be time varying. We make the assumption that w_k and v_k are independent, white Cauchy distributed random variables. It was shown in [4] that the characteristic function of the ucpdf at any step k given a scalar realization of the measurement sequence history $y_k = \{z_1, z_2, ..., z_k\}$ can be expressed as

$$\bar{\phi}_{X_k|Y_k}(\nu) = \sum_{i=1}^{N_t^{k|k}} g_i^{k|k}\left(y_{gi}^{k|k}(\nu)\right) \exp\left(y_{ei}^{k|k}(\nu)\right), \tag{2}$$

where X_k and Y_k denotes the random state vector and measurement history, respectively, and $\nu \in \mathbb{R}^d$ is the spectral vector. The coefficients' function $g_i^{k|k}\left(y_{gi}^{k|k}(\nu)\right)$ is given by

$$g_i^{k|k}\left(y_{gi}^{k|k}(\nu)\right) = \frac{1}{2\pi}\left[\frac{g_{r_i^{k|k}}^{k-1|k-1}\left(y_{gi1}^{k|k}(\nu) + h_i^{k|k}\right)}{jc_i^{k|k} + d_i^{k|k} + y_{gi2}^{k|k}(\nu)} - \frac{g_{r_i^{k|k}}^{k-1|k-1}\left(y_{gi1}^{k|k}(\nu) - h_i^{k|k}\right)}{jc_i^{k|k} - d_i^{k|k} + y_{gi2}^{k|k}(\nu)}\right], \tag{3}$$

where

$$y_{gi}^{k|k}(\nu) = \left(q_i^{k|k}\right)^T \lambda_i^{k|k}(\nu) \in \mathbb{R}, \tag{4a}$$
$$\lambda_{il}^{k|k}(\nu) = \operatorname{sgn}\left(\left\langle a_{il}^{k|k}, \nu\right\rangle\right),\ l \in [1, ..., m_i^{k|k}],\ \lambda_i^{k|k} \in \mathbb{R}^{m_i^{k|k}}, \tag{4b}$$
$$y_{ei}^{k|k}(\nu) = -\sum_{l=1}^{m_i^{k|k}} p_{il}^{k|k}\left|\left\langle a_{il}^{k|k}, \nu\right\rangle\right| + j\left\langle b_i^{k|k}, \nu\right\rangle \in \mathbb{C}. \tag{4c}$$

These expressions must be unpacked for clarity. Throughout the paper we refer to a '*term*' of the characteristic function as any matrix, vector or scalar with subscript $(\cdot)_i$, and $i \in \left[1, 2, \ldots, N_t^{k|k}\right]$, with $N_t^{k|k}$ defining the number of terms in the characteristic function at step k, given measurement sequence y_k. Formally, a superscript $(\cdot)^{k|k-1}$ or $(\cdot)^{k|k}$ indicates that the term was computed before or after the most recent measurement z_k was processed, respectively.

As derived in [4], the characteristic function in (2) and its first two derivatives are evaluated using (3) and (4) as $\nu \to 0$ and at any point denoted by $\bar{\nu}$ (minor restriction on $\bar{\nu}$ apply: see [4] for details) to generate the conditional mean $\hat{x}_k$ and the error variance P_k at each estimation step k, given by the expressions:

$$\hat{x}_k = \frac{1}{j f_{Y_k}} \sum_{i=1}^{N_t^{k|k}} g_i^{k|k}(\bar{\nu}) \bar{y}_{ei}^{k|k}(\bar{\nu}) \in \mathbb{R}^d, \tag{5}$$

$$P_k = -\frac{1}{f_{Y_k}} \sum_{i=1}^{N_t^{k|k}} g_i^{k|k}(\bar{\nu}) \bar{y}_{ei}^{k|k}(\bar{\nu}) \left(\bar{y}_{ei}^{k|k}(\bar{\nu})\right)^T - \hat{x}_k \hat{x}_k^T \in \mathbb{R}^{d\times d}, \tag{6}$$

where

$$f_{Y_k} = \sum_{i=1}^{N_t^{k|k}} g_i^{k|k}(\bar{\nu}) \in \mathbb{R}, \tag{7}$$

$$\bar{y}_{ei}^{k|k}(\bar{\nu}) = -\sum_{l=1}^{m_i^{k|k}} p_{il}^{k|k} \lambda_{il}^{k|k}(\bar{\nu}) a_{il}^{k|k} + j b_i^{k|k} \in \mathbb{C}^d. \tag{8}$$

Equation (3), derived in [4], is a recursively growing function, requiring all coefficients generated from terms at steps 1 though k. Here, $c_i^{k|k} \in \mathbb{R}$, $d_i^{k|k} \in \mathbb{R}$, $y_{gi1}^{k|k} \in \mathbb{R}^{k-1}$, $y_{gi2}^{k|k} \in \mathbb{R}$, $h_i^{k|k} \in \mathbb{R}^{k-1}$, $q_i^{k|k} \in \mathbb{R}^{m_i^{k|k}}$, $a_{il}^{k|k} \in \mathbb{R}^d$, $b_i^{k|k} \in \mathbb{R}^d$, $p_i^{k|k} \in \mathbb{R}^{m_i^{k|k}}$. The variables listed above represent all *elements* of a single *term* i of the characteristic function at step k given the measurement history y_k. The elements $c_i^{k|k}$, $d_i^{k|k}$ and $q_i^{k|k}$ are used to store all generated coefficients from step 1 to k, and are needed when 'tunneling' back to step 1 in (3). Each term i has several coefficient vectors a_{il} that are needed when forming (4c) and (3), where the number of coefficient vectors per term i is stored as $l \in \left[1, \ldots, m_i^{k|k}\right]$. The elements a_{il} are used extensively in the development of the reformulated estimator in section III. We note that when a superscript is omitted (*e.g.*, $(\cdot)_i$), the reader can assume the term has been formed after the most current measurement $(\cdot)_i^{k|k}$.

The formulation shown in (3) for computing $g_i^{k|k}(y_{gi}(\nu))$ (shorthand $g_i(\nu)$) unfortunately must tunnel the new arguments at step $k|k$ recursively through each numerator function $g_{r_i^{k|k}}^{k-1|k-1}(\cdot)$ until they have reached the first step, and then back up to step k. Effectively, $g_{r_i^{k-1|k-1}}$ is composed of a hierarchy of divisions ending at k-1 with a divisor similar to that in (3) applied at k and going up to $k = 1$. This is costly, however, as at each successive estimation step the computation and memory requirements to compute a

single $g_i^{k|k}(\nu)$ increases. In addition, the formulation presented in this section does not possess a structure that is able to combine terms with redundant parameterizations, as terms generated at past estimation steps k were required in memory to evaluate the recursive $g_i^{k|k}(\nu)$ expression. A reformulation of $g_i^{k|k}(\nu)$ in section III-B resolves these issues.

III. Reformulated Multivariate Cauchy Estimator

Many terms of the characteristic function contain arguments of the exponential $y_{ei}(\nu)$ that are the same, but coefficients $g_i(\nu)$ that are different. If the construction of (3) could be formulated such that terms were additive with one another, a large number of terms of the characteristic function could be combined together. An examination of the g-function in (3), (4a) and (4b) shows that its values are constant within certain regions. Explicitly, (4b) is a function of the spectral variable ν, which constructs sign values over the inner-product of a_{il} and ν. These sign values are the arguments to (3) and remain constant for certain regions of ν. It is also noticed that the a_{il} coefficients operating on the spectral variable ν form a central arrangement of hyperplanes in the spectral domain $\mathbb{R}^d$. The set of vectors $a_{il}, l \in \left[1,\dots,m_i^{k|k}\right]$, grouped into a matrix $A_i \in \mathbb{R}^{m_i^{k|k}\times d}$, defines a central arrangement of hyperplanes in $\mathbb{R}^d$, where $m_i^{k|k}$ denotes the number of hyperplanes in the arrangement of the term i.

Using this alternative viewpoint of the vectors a_{il}, the sign values $\operatorname{sgn}\left(a_{il}^T\nu\right), l \in \left[1,\dots,m_i^{k|k}\right]$ can then be thought of as forming a *sign-vector*, which uniquely describes a *cell* of the hyperplane arrangement. A cell can be described by a sequence of ±1 values, or sign-vector, where a ±1 indicates the point (*e.g.,* ν) being tested lies in the positive (or negative) halfspace of a particular hyperplane, respectively. The cells of a central arrangement form separate, unbounded, convex regions [9]. Due to the observations made regarding (4b), the function $g_i(\nu)$ evaluates to a *constant* in every point of a cell. If one could find the set of sign-vectors that describe all cells of an arrangement, then, as shown in [10], a basis can be constructed using these sign-vectors associated with each cell, such that the numerator of (3) can be represented as a linear combination with the elements of this basis. In this section, we reformulate the characteristic function of section II into a computationally reduced form, by constructing the novel basis of [10].

A. Centralized Hyperplane Arrangements

As shown in [4], terms of the characteristic function are updated twice per estimation step k: once in the 'time-propagation' (TP) routine and again in the 'measurement update' (MU) routine. Of all the elements per term i presented in section II, the centralized hyperplane arrangement of the elements a_{il}, grouped into A_i, is the most crucial to the development of the reformulated estimator. The time-propagation function is responsible for updating the hyperplane arrangements A_i from k-1$|k$-1 to $k|k$-1 and is given by

$$A_i^{k|k-1} = \begin{bmatrix} A_i^{k-1|k-1}\Phi^T \\ \Gamma^T \end{bmatrix} \in \mathbb{R}^{(m_i^{k-1|k-1}+r)\times d}. \tag{9}$$

Note that newly 'propagated' hyperplanes are a transformation of the *parent* hyperplane arrangement by the dynamics matrix Φ^T and concatenated to Γ^T. This forms a new arrangement of $m_i^{k|k-1} = m_i^{k-1|k-1} + r$ hyperplanes for each term i, where r is equal to the number of columns of Γ (minor restrictions apply, see section V-A).

Once a new measurement z_k is obtained, the newly propagated parent hyperplanes $A_i^{k|k-1}$ are updated by the measurement update function. MU is responsible for generating *child* arrangements $A_{(\cdot)}^{k|k}$. Given $m_i^{k|k-1}$ hyperplanes for arrangement $A_i^{k|k-1}$, term i produces $m_i^{k|k-1}+1$ child hyperplane arrangements $A_j^{k|k}$ according to

$$\begin{aligned} \mu_h &= A_i^{k|k-1}H^T \in \mathbb{R}^{m_i^{k|k-1}}, \\ \mu_{il}^{k|k-1} &= \begin{cases} A_{il}^{k|k-1}\cdot\frac{1}{\mu_{hl}}, & \text{if} \quad l \in [1,...,m_i^{k|k-1}] \\ \mathbf{0}^{1\times d}, & \text{if} \quad l = m_i^{k|k-1}+1 \end{cases}, \\ \mu_i &\in \mathbb{R}^{(m_i^{k|k-1}+1)\times d}, \\ A_{jl}^{k|k} &= \mu_{il}^{k|k-1} - \mu_{ij}^{k|k-1}, \quad A_j^{k|k} \in \mathbb{R}^{m_i^{k|k}\times d}, \\ & j \in [1,...,m_i^{k|k}+1], \quad l \in [i,...,m_i^{k|k}+1], j \neq l. \end{aligned} \tag{10}$$

Note that $N_t^{k|k-1} = N_t^{k-1|k-1}$ after TP and $m_i^{k|k} = m_i^{k|k-1}$ after MU. In total, $N_t^{k|k-1}$ parent terms create $N_t^{k|k} = \sum_{i=1}^{N_t^{k|k-1}}\left(m_i^{k|k-1}+1\right)$ child terms after the measurement update. The updated arrangements at $k|k$ are formed by scaled differences of the propagated hyperplanes. Due to the $j \neq l$ exclusion in (10), the new arrangements at $k|k$ have the same number of hyperplanes as step $k|k$-1. Note that the $A_{jk}^{k|k}$ hyperplanes becomes zero when $j = l$, Once all child terms are generated, all children are re-indexed continuously as $i = \left[1,2,\dots,N_t^{k|k}\right]$. The propagated parent arrangements $A_i^{k|k-1}$ and child arrangements $A_i^{k|k}$ are now used to reformulate (3).

B. Reformulated g Coefficient-Functions

In [10], the authors show that if a function $g(\nu) \in \mathbb{C}$ is constant when evaluated at any point within a cell of a hyperplane arrangement, then a relationship between the value of the function at ν, *i.e,* $g(\nu) \in \mathbb{C}$, and a *basis-vector* $S(\lambda(\nu)) \in \mathbb{R}^{s(m,d)}$ can be formed as

$$g(\nu) = S^T(\lambda(\nu))\,\alpha, \;\; g(\nu) \in \mathbb{C}, \;\; \alpha \in \mathbb{C}^{s(m,d)}, \tag{11}$$

$$\lambda(\nu)_j = \operatorname{sgn}\left(\langle a_j, \nu\rangle\right), \;\; j \in [1,...,m], \; a_j \in A, \tag{12}$$

$$s(m,d) = \sum_{k=0}^{d}\binom{m}{k}, \tag{13}$$

where $S(\lambda(\nu)) : \mathbb{R}^m \to \mathbb{R}^{s(m,d)}$ denotes the expansion to generate a basis-vector from a sign-vector $\lambda(\nu)$ of a cell located at ν. m denotes the number of hyperplanes in the arrangement and the number of sign values in the sign-vector $\lambda(\nu)$. $s(m,d)$ is the size of the basis-vector $S(\cdot)$, $a_j \in \mathbb{R}^d$ are the hyperplane coefficients of the central hyperplane arrangement $A \in \mathbb{R}^{m\times d}$ and α is a complex valued vector, which parameterizes the relationship between the value of $g(\cdot)$ that is constant in a cell and its corresponding basis-vector $S(\cdot)$. $\lambda(\nu)$ denotes the sign-vector ($\{\pm1\}^m$) of length m, which is uniquely defined for each cell of a hyperplane arrangement.

As described in [10], the function $S(\lambda(\nu))$ combinatorially expands a sign-vector $\lambda(\nu)$ (defined at ν) up to products of d combinations. For example, the sign-vector of a three hyperplane arrangement in three dimensions, symbolically, is $\lambda(\nu) = [\sigma_1, \sigma_2, \sigma_3]$ at any ν, where $\sigma \in \{-1, 1\}$. The basis-vector is then formed by combinatorially expanding the sign-vector $\lambda(\nu)$ as $S(\lambda(\nu)) = [1,\ \sigma_1, \sigma_2, \sigma_3,\ \sigma_1\sigma_2, \sigma_1\sigma_3, \sigma_2\sigma_3,\ \sigma_1\sigma_2\sigma_3]$, to length $s(3,3) = 8$. Note this operation will be performed row-wise for matrix arguments to $S(\cdot)$ in the following paragraphs.

In order to determine α, one must compute the sign-vector $\lambda(\nu)$ and constant value $g(\nu)$ for every cell of an arrangement. Explicitly, we refer to the set of all sign-vectors that uniquely defines each

cell of a hyperplane arrangement $A_i \in \mathbb{R}^{m_i \times d}$ as the *enumeration matrix* $B_i \in \mathbb{R}^{c(A_i)\times m_i}$, for term i, where m_i denotes the number of hyperplanes in term i and $c(A_i)$ is the number of cells in A_i. That is, B_i stores $c(A_i)$ sign-vectors of length m_i row-wise, where each sign-vector uniquely locates a particular cell in the arrangement. In section IV, we show explicitly how this matrix can be formed, but here we assume the matrix B_i has already been determined for A_i. The details for locating the constant value in every cell are given in section V.

If one can compute the enumeration matrix B_i for the hyperplane arrangement of each term, A_i, then the vector α can be computed by solving an under-determined set of linear equations (i.e., a system of $c(A_i)$ equations (11)). A least squares solution to this system for the measurement updated term i at k-1$|k$-1 is given by

$$\alpha_i^{k-1|k-1} = S^{\dagger}(B_i^{k-1|k-1})\bar{g}_i^{k-1|k-1}. \tag{14}$$

Here, $S(B_i) \in \mathbb{R}^{c(A_i)\times s(m_i,d)}$, with $c(A_i) \leq s(m_i, d)$, is the *basis matrix*, which generates the $c(A_i)$ basis-vectors of length $s(m_i, d)$ (stored row-wise in $S(B_i)$) from a combinatorial expansion of the $c(A_i)$ sign-vectors of length m_i that are stored row-wise in the enumeration matrix B_i. The basis-vectors (rows) of $S(B_i)$ define a basis for α_i over the $c(A_i)$ cells of A_i. $\bar{g}_i \in \mathbb{R}^{c(A_i)}$ is a vector container which stores all $c(A_i)$ constant values of $g_i(\nu)$ in (3). In (14), $S^{\dagger}(B_i)$ is the pseudo-inverse of $S^T(B_i)$. Note that all the variables discussed above have the timing index (superscript) k-1$|k$-1, omitted here for brevity. Consequently, following (11), the numerator coefficient function $g_i^{k-1|k-1}(\nu)$ of (3) can now equivalently be formed by using $\alpha_i^{k-1|k-1}$ with an appropriate basis vector.

Next, we address the time propagation of $\alpha_i^{k|k-1}$ that will be used to construct the time propagated $g_i^{k|k-1}(\nu)$. As shown in (9), the hyperplanes of each term are rotated by Φ^T and additional hyperplanes, defined by Γ^T, are added to the hyperplane arrangement. TP, *i.e.*, (9), changes the orientation of the halfspaces of the propagated parent hyperplanes $A_i^{k|k-1}$with respect to the parent arrangement $A_i^{k-1|k-1}$. One must update $\alpha_i^{k-1|k-1}$ after the TP routine, based on how the dynamics matrix Φ^T has rotated the parent arrangement $A_i^{k-1|k-1}$ in TP, and thus, has changed the sign-vector defined at ν for the propagated parent hyperplanes $A_i^{k|k-1}$.

The relation between the parent $\alpha_i^{k-1|k-1}$ and the propagated $\alpha_i^{k|k-1}$ can be expressed compactly by using the Hadamard product $\circ$, which denotes element-wise multiplication of two vectors. It is given by

$$\alpha_i^{k|k-1} = S\left(\hat{\lambda}_i^{k|k-1}(\nu)\right) \circ \alpha_i^{k-1|k-1}, \tag{15}$$

$$\hat{\lambda}_i^{k|k-1}(\nu) = \operatorname{sgn}\left(A_i^{k|k-1}[1 : m_i^{k|k-1} - r, :] \cdot \nu\right) \in \mathbb{R}^{m_i - r}. \tag{16}$$

We use brackets $[\cdot, \cdot]$ to denote only the first $m_i^{k|k-1} - r$ rows (or a 'partition') of the arrangement $A_i^{k|k-1}$ are accessed. The hyperplane arrangement is partitioned because Γ^T is appended to the arrangements in the TP routine at step k and is unknown to the parent arrangement formed at previous step k-1. In (16), sgn() is an element-wise sign operation over each entry of its input vector. In addition, the $\hat{(\cdot)}$ symbol indicates that the variable was formed through a partition of the hyperplanes.

Now we are ready to address the measurement update to determine $g_i^{k|k}(\nu)$. Using the new representation introduced in this section, (3) and (4) can be restated as

$$g_i^{k|k}(\nu) = \frac{1}{2\pi}\left[\frac{S_{\psi_i}^{+}\left(\hat{\lambda}_i^{k|k}(\nu)\right)^T \alpha_i^{k|k-1}}{jc_i + d_i + y_{gi}} - \frac{S_{\psi_i}^{-}\left(\hat{\lambda}_i^{k|k}(\nu)\right)^T \alpha_i^{k|k-1}}{jc_i - d_i + y_{gi}}\right], \tag{17}$$

where

$$g_i^{k|k}(\nu) \in \bar{g}_i^{k|k}, g_i^{k|k} \in \mathbb{C}, \bar{g}_i^{k|k} \in \mathbb{C}^{c(A_i)}, c_i, d_i, y_{gi} \in \mathbb{R},$$
$$S_{\psi_i}^{+}\left(\hat{\lambda}_i^{k|k}(\nu)\right)^T \alpha_i^{k|k-1} = \bar{g}_{r_i^{k|k}}^{k-1|k-1}\left(y_{gi1}^{k|k}(v) + h_i^{k|k}\right),$$
$$S_{\psi_i}^{-}\left(\hat{\lambda}_i^{k|k}(\nu)\right)^T \alpha_i^{k|k-1} = \bar{g}_{r_i^{k|k}}^{k-1|k-1}\left(y_{gi1}^{k|k}(v) - h_i^{k|k}\right), \tag{18}$$

$$\hat{\lambda}_i^{k|k}(\nu) = \operatorname{sgn}\left(A_i^{k|k}[1 : m_i - r, :] \cdot \nu\right) \in \mathbb{R}^{m_i - r},$$
$$\lambda_i^{k|k}(\nu) = \operatorname{sgn}\left(A_i^{k|k} \cdot \nu\right) \in \mathbb{R}^{m_i}, \tag{19}$$

$$y_{gi} = q_i^T \cdot \lambda_i^{k|k}(\nu) \in \mathbb{R}, \quad q_i \in \mathbb{R}^{m_i}. \tag{20}$$

The procedure to form the basis-vectors $S_{\psi_i}^{\pm}\left(\hat{\lambda}_i^{k|k}(\nu)\right)$ is elaborated upon. Due to the $j \neq l$ exclusion in (10), the sign values stored in the child sign-vector $\lambda_i^{k|k}(\nu)$ are not aligned correctly with respect to the indexing used to store the sign values in the parent sign-vector $\lambda_i^{k-1|k-1}(\nu)$. In essence, the indexing scheme used to store hyperplanes in the child arrangement during MU (step $k|k$) is inconsistent with respect to how the parent term at k-1$|k$-1 has stored its hyperplane arrangement, and thus its sign-vector as well. This makes the process of correctly forming the basis-vector to be used in (17) slightly involved. The notation $S_{\psi_i}^{\pm}(\cdot)$ is a useful syntax that denotes the procedure to re-index the sign-vector with respect to its parent and form the appropriate basis-vector for (17). The subscript $(\cdot)_{\psi_i}$ informs the basis-vector that either a ± 1 should be inserted at the ψ_i-th index in the sign-vector $\hat{\lambda}(\nu) \in \mathbb{R}^{m_i}$. The superscript $(\cdot)^{\pm}$ informs the basis-vector whether a ± 1, respectively, should be inserted at this index. This is detailed in the novel integration formula given in [4], where the authors explain the left-side fraction of (3) assigns a $+1$ value to indices in the sign-vector where the exclusion occurs, while the right-side fraction assigns a -1 to this position. It should be noted that after $S_{\psi_i}^{\pm}(\cdot)$ inserts a ± 1 at index ψ_i to the sign-vector, the m_i-lengthed vector would now be of temporary length $m_i + 1$. The $m_i + 1$-th element is not required and is clipped to form a new m_i-lengthed vector. The clipped sign-vector is then combinatorially expanded to length $s(m_i, d)$ as detailed previously, which generates the basis-vectors for both numerators of (17).

It is important to note the following consequential aspects of this new formulation. Firstly, the term $q_i \in \mathbb{R}^{m_i}$ no longer stores all past steps q-values, but is now a vector of solely coefficients generated at the current step. Second, the scalar elements c_i and d_i of past steps (*e.g,* k-1) are now unneeded. This is due to the fact that new arguments no longer must 'tunnel' backwards to previous steps through the numerators of (3). And finally, the form of the g-function in (17) is now additive and terms with equivalent arguments of the exponential y_{ei} can be combined, as is detailed in section V-B.

IV. Enumeration of Central Arrangements

As illustrated in Figure 1, cell enumeration algorithms for hyperplane arrangements can be categorized into two classes: general or central. While two arrangements (either general or central) may contain the same number of hyperplanes, various geometrical patterns can create degeneracies and reduce the total number of

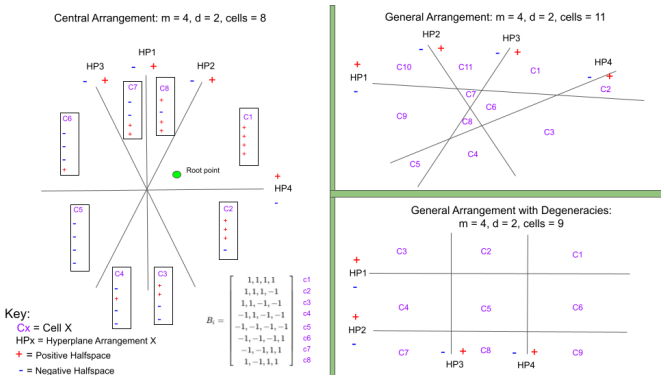

Fig. 1. Enumeration examples of 4 hyperplanes in 2 dimentions for both general and central arrangements. Left: enumeration of a central arrangement with illustrated hyperplane half-spaces and cell sign-vectors. Top Right: Non-degenerate general arrangement with 11 cells. Bottom Right: Degenerate general arrangement (tic-tac-toe board) with 9 cells.

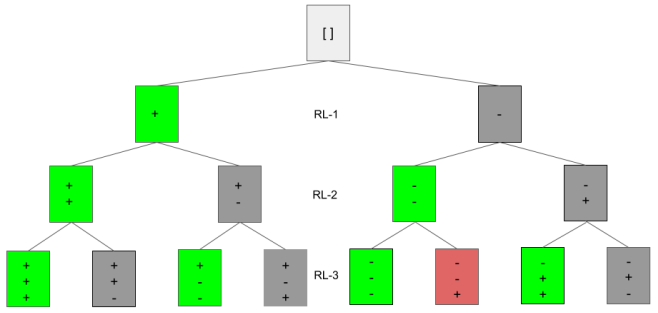

Fig. 2. Incremental enumeration (Inc-Enu) tree for cell enumeration of a four hyperplane, three dimentional arrangement. Boxes in gray denote an LP was solved to validate the current sign sequence. Boxes in green denote no LP was needed to validate the sign sequence. Boxes in red denote the LP was infeasible and the proposed sign sequence is invalid. RL denotes the recursion level the sequence is formed at.

cells in the arrangement. It becomes imperative then that a cell enumeration algorithm can efficiently identify these degeneracies. This is illustrated in the two dimensional example in Figure 1 (right), where we see a tic-tac-toe board has less cells than a general arrangement with non-parallel hyperplanes.

We focus our attention to cell enumeration for central arrangements, since they characterize the characteristic function of the Cauchy Estimator. In the following sub-sections, we first provide a high level overview of Incremental-Enumeration (Inc-Enu), an efficient cell enumeration algorithm for hyperplane arrangements, followed by how this algorithm is parallelized on the GPU for our real-time Cauchy Estimator. We refer the reader to [8] for a detailed discussion of cell enumeration and reverse-search methods.

A. *Sequential Cell Enumeration*

The goal of cell enumeration for hyperplane arrangements is to compute the set of sign-vectors which uniquely defines every cell. For central arrangements, these sign vectors can be found by solving Phase-I feasibility linear programs (LPs) [9], explained in detail by the authors of Inc-Enu in [7]. Naively, one could attempt 2^m feasibility LPs to test all permutations of sign-vectors $\{\pm 1\}^m$, given an arrangement with m-hyperplanes. This strategy becomes highly intractable, however, as m grows larger than 10. Cell enumeration algorithms, instead, solve a sequence of LPs, gaining insight after each successful solve into which cells can be found next. If the LP is found infeasible, the sign-vector attempted is known not to belong to any given cell in the arrangement.

Figure 2 depicts the enumeration scheme 'Incremental-Enume–ration' (*Inc-Enu*) presented in [7]. Inc-Enu is a recursive, depth-first-search algorithm. The algorithm starts at the root of the tree in fig. 2 with knowledge of the root-point (the first known feasible point) and the symmetric-generator. The goal of Inc-Enu is to recursively build up sign-vectors for each cell through sequentially solving LPs. At each recursion level j, Inc-Enc must determine whether an incrementally growing sign-sequence of length j remains a valid cell identifier. If the sign-sequence is found valid at all recursion levels, the sequence is deemed a sign-vector of a cell in the arrangement. In cell enumeration, it is common to refer to the LP solver as an *oracle* who answers *queries*.

If a given sign sequence is found infeasible (shown in red) by the oracle, Inc-Enu will pop up a recursion level instead of recuring down to depth $m-1$ to form the new sign-vector. We note that any sign sequences that are found infeasible by the oracle do not have left and right children and therefore are leaves of the tree. We see that 'left' child nodes (shown in green) of the parent are found without querying the oracle, whereas 'right' children (shown in red/black) denote where proposed sign sequences are assigned to the the oracle for query.

We see that the enumeration in Figure 2 yields 7 feasible sign-vectors in the positive halfspace of the symmetric generator, implying there are 14 total cells in the (non-degenerate) arrangement. Note that the maximum number of cells is 14 for a four hyperplane, three dimensional central arrangement [8].

B. *Parallel Incremental Enumeration*

Here, we detail the minor but crucial changes made to the Inc-Enu routine to parallelize the algorithm for real-time performance. As our proposed strategy uses a combination of sequential and parallel compute operations, we coin the method 'Hybrid Inc-Enu' (HIE).

While Inc-Enu conducts a depth-first search, HIE is a parallel breadth-first search reformulation that operates on a forest (many trees) of enumeration tasks, solving a grid of computation at each sequential step. Figure 3 illustrates this parallel enumeration process. To begin, HIE is given N arrangements of m-hyperplanes to be enumerated in dimension d. That is, HIE takes the results of the first grid at step 1 (denoted S1) as input to solve the next grid, S2. Note that all enumeration tasks located within a given step (or depth) of the forest are independent of one another and can be solved in parallel. We see that at every step of parallel computation, CUDA blocks are assigned to the active enumeration tasks of each tree. Each block produces a left and right child (also CUDA blocks), if and only if the block's proposed sign-sequence is found valid. The left child (seen in green) is provided with both the sign-sequence and computed feasible point of its parent, and simply needs to append which halfspace the known feasible point lies in with respect to the next hyperplane to be enumerated. Left children are computationally cheap, as these blocks do not need to solve an LP. Each right-child block (seen in gray/red) is given the validated parent sign-sequence concatenated to the *opposite* of the sign computed by the left child, as input. The right-child is responsible then for solving a feasibility LP to validate this proposed sign-sequence. This implies each block must have its own queryable oracle to assert the validity of sign-sequences.

Programatically, this translates to developing a parallel Phase-I simplex method [9], where each CUDA block is responsible

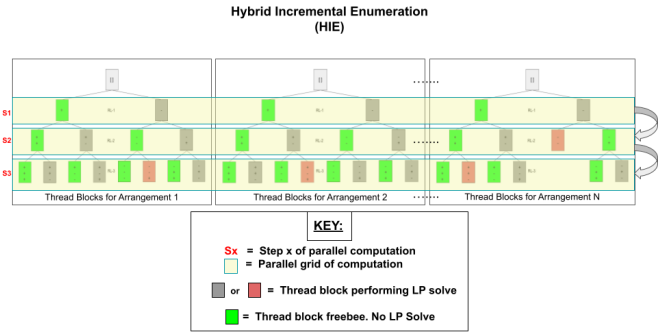

Fig. 3. Hybrid Inc-Enu example forest, with N arrangements of m-hyperplanes in dimention d. Blocks (thread blocks) within the grid at each step are solved in parallel. Red/Black boxes indicate a CUDA block solved an LP. Green boxes indicate no LP was needed to validate the sign-sequence.

for computing a simplex tableau (LP) of a single, proposed sign-sequence. Previous works on GPU linear programming [11] focus their attention on using the GPU-compute resources to parallelize tableau operations for individual large and/or sparse LPs. Instead, we focus the GPU-compute resources on efficiently parallelizing many *small* Phase-I LPs for throughput. The authors in [12] propose a Phase I and Phase II simplex method for simultaneously solving small batched LPs. Here, we develop HIE to streamline the inputs and outputs of the batched Phase-I simplex algorithm proposed in [12] explicitly for solving batched cell enumerations.

V. Flattening

Here we construct the vectors α_i per term that are needed in (14) and (17) and consequently show how the coefficient function $g_i^{k|k}(\nu)$ is expressed using the new basis. This process is referred to as *flattening*, as it eliminates the need to hierarchically store terms of the characteristic function from past steps. As mentioned in section III-B, the $c(A_i)$ constant cell values $g_{ij}^{k|k} \in \mathbb{C}$, $j \in [1, 2, \ldots, c(A_i)]$ are stored in the vector $\bar{g}_i^{k|k} \in \mathbb{C}^{c(A_i)}$, $g_{ij}^{k|k} \in \bar{g}_i^{k|k}$ $\forall j$ and are needed to determine $\alpha_i^{k|k}$. The constant value located in every cell of an arrangement can be explicitly calculated by using each row of the enumeration matrix B_i, *i.e,* the sign-vector of a particular cell, to generate the appropriate basis-vectors for all $c(A_i)$ cells.

We restate the expressions in (17) to (20), while replacing the sign vector $\hat{\lambda}^{k|k}(\nu)$ by the first $m_i^{k|k} - r$ elements of each j-th row of the enumeration matrix B_i, i.e.,

$$g_{ij}^{k|k} = \frac{1}{2\pi}\left(\frac{S_{\psi_i}^{+}\left(\hat{\beta}_j\right)^T \alpha_i^{k|k-1}}{jc_i + d_i + y_{gi}} - \frac{S_{\psi_i}^{-}\left(\hat{\beta}_j\right)^T \alpha_i^{k|k-1}}{jc_i - d_i + y_{gi}}\right), \quad (21)$$

$$\hat{\beta}_j = B_i\left[j, 1 : m_i^{k|k} - r\right], y_{gi} = q_i^T B_{ij}, j \in [1, 2, \ldots, c(A_i)].$$

Just as in (17), only the first $m_i^{k-1|k-1} = m_i^{k|k} - r$ elements of the sign-vectors stored row-wise in B_i are given to $S_{\psi_i}^{\pm}(\cdot)$. The vector $\alpha_i^{k|k}$ is found by the least-square solution in (14).

A. Coaligning Hyperplanes

It was mentioned in section III-A that the relationship $m_i^{k|k} = m^{k|k-1} = m^{k-1|k-1} + r$ has a slight caveat, which directly impacts the flattening routine. As noted in [4], it is possible for the TP/MU routines to generate arrangements with redundant (or coaligned) hyperplanes. We refer to the procedure of removing hyperplanes from an arrangement whose normal vectors are parallel as *coalignment*. Coalignment is presented in [4], where the authors detail a procedure to handle all the elements of a term when coaligned hyperplanes occur. Coalignment is necessary for two reasons. First, HIE would fail to locate cells between two redundant hyperplanes in an arrangement, or may falsely locate cells due to numerical round-off error in the hyperplanes. Second, (10) used by MU would create singular hyperplanes when forming the child arrangements.

B. Combining Terms

Terms that have equal parameterizations of their exponential argument y_{ei} can be combined by summation of the respective $\alpha_{(\cdot)}$ vectors. Specifically, y_{ei} in (4c) is parameterized by a_{il}, replaced in the new formulation by A_i, p_{il} and b_i. If all the parameters of terms i and j are equal within a small numerical ϵ value, i.e., $|A_{ilk} - A_{jlk}| \leq \epsilon$, $|p_{il} - p_{jl}| \leq \epsilon$, and $|b_{ik} - b_{jk}| \leq \epsilon$, for all $l \in \left[1, \ldots, m_i^{k|k}\right]$ and $k \in [1, \ldots, d]$, these two (or more) terms can be combined. To save computational effort, the comparisons above can be restricted to the $i < j$ condition. The process of combining terms, although requiring additional processing, was empirically shown to reduce the computational burden of the estimator tremendously, as is shown in section VI.

VI. Experiments, Timing Results, Extensions

In this section, we showcase the performance of the Cauchy Estimator against the Kalman Filter for a three-state linear system in a Cauchy and Gaussian noise simulation. We discuss the significance reducing terms has on the computational burden of the estimator. We report the real-time execution rates for estimates conditioned on six, seven and eight measurements, using an NVIDIA GeForce GTX 1060 GPU for CUDA support to run the proposed estimation scheme in parallel. This approach is motivated by our future work to generalize the sliding, fixed measurement window technique that was used previously in the two-state estimator [6] to any dimension. This approach limits the computation of the estimator at each sample time step by taking estimates only conditioned on the last W measurements. By running this technique simultaneously on W sliding windows, real-time performance can be achieved for large time scales, with a single estimate conditioned on the last W measurements provided at each step.

The performance results of the Cauchy Estimator presented in this section, while insightful, are not our novel contribution, as similar insights on the performance of the Cauchy Estimator when compared to the Kalman Filter have previously been made in [6, 13] for two-state problems. Our contribution here is that we show how the reformulated Cauchy Estimator can achieve the same simulation performance as in [4, 6, 13], but for multivariate systems and with a remarkably reduced number of terms encompassing the characteristic function at each step k. Furthermore, as all terms are independent of one another, parallel programming is seen to be highly advantageous.

We compare the performance of the two estimators in both a Cauchy and Gaussian noise simulation using the system dynamics proposed in [4]. We set the statistics of the Kalman Filter and our reformulated Cauchy Estimator equal as proposed in [13], through a non-linear least-squares parameter fit of the two pdfs. The Cauchy Estimator was not explicitly compared to the Kalman Filter in [4] and is presented here for multivariate systems.

Figure 4 (left column) illustrates the interesting properties of the Cauchy Estimator in a Cauchy noise simulation. As given in (6),

we see that the variance of the Cauchy Estimator is explicitly a function of the measurement, and therefore the conditional variance dynamically adjusts to the measurement history. This is not the case in the Kalman Filter, where the filter's posterior covariance can be calculated a-priori. At $k = 3$, a pulse occurs in the process noise. We see that the Cauchy Estimator tracks this jump in all three states with ease, while the Kalman Filter mostly ignores this pulse in states one and three, producing large state errors. At $k = 4$, a pulse occurs in the measurement noise. We see the Cauchy Estimator ignores this pulse while sharp jumps are seen in the Kalman Filter.

Figure 4 (right column) shows the performance of both estimators in Gaussian noise simulation. We see clearly, in Gaussian noise, the Kalman Filter is the superior estimation scheme. It is interesting to note how the one-sigma values of the Cauchy Estimator upper-bound those of the Kalman Filter and that in Gaussian noise, both of the estimators one-sigma values bound the estimation error at each step.

Reducing terms, while a computationally expensive procedure, is seen to have a tremendous computational advantage over not reducing terms at the end of each estimation step k. Table I illustrates the savings. At step $k = 8$, the new estimation scheme encompasses only one-percent of the former number of terms required. Table II shows the performance of six, seven, and eight measurement windows for the three-state problem given. Execution times for three-state dynamic systems would follow those of Table II for both time-varying and time-invariant systems. For the windowing process suggested earlier for the time-invariant case, each W-window will produce the same parameter sequence with the same initial conditions. Storing these parameters yields dramatic improvement in run time over that shown in Table II. The execution speed and hertz-rates (hz-rates) of Table II indicate the applicability of the Cauchy Estimator for engineering applications.

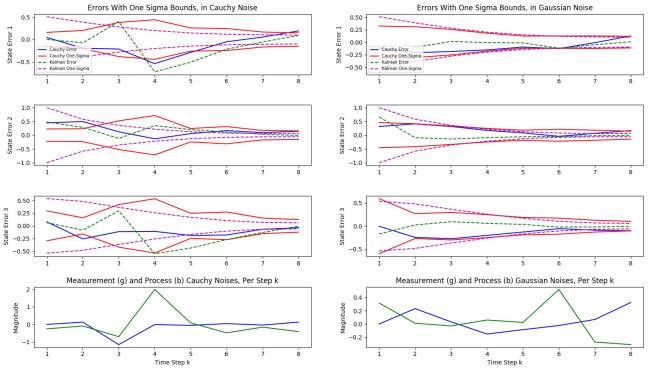

Fig. 4. Comparison of Cauchy Estimator to Kalman Filter in Gaussian and Cauchy noise environments. Cauchy errors are primarily bounded by their one-sigma values in Cauchy-noise, whereas both estimators are bounded by their one-sigma values in Gaussian noise.

TABLE I
NUMBER OF TERMS ELIMINATED BY THE TERM REDUCTION ALGORITHM AT EACH ESTIMATION STEP. DEPICTED ARE THE RESULTS OF REDUCING TERMS FOR THE THREE-STATE EXAMPLE GIVEN.

Step Counter	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 6$	$k = 7$	$k = 8$
No Term Reduction	4	20	120	792	5,440	37,936	216,066	>1.69M
With Term Reduction	4	14	48	161	542	1,762	5,709	18,594

TABLE II
REAL-TIME WINDOW PERFORMANCE ON NVIDIA GEFORCE 1060X GPU, SINGLE-STREAMED

Window Size	Total Execution Time (sec)	Window Hz-Rate
6	0.015	66.67
7	0.053	18.87
8	0.352	2.84

VII. CONCLUSIONS

A reformulation of the Cauchy Estimator in [4] is presented in a computationally reduced form. It was shown that enumerating the cells of the central hyperplane arrangements of the characteristic function 'flattened' the numerator of (3) by constructing the basis of [10]. Moreover, the reformulated g-function is additive and terms can combine. CUDA-support was shown to allow the estimator to run at real-time rates. In future work, we aim to present several applications to demonstrate the utility of the real-time estimator in volatile noise environments. Extensions to nonlinear systems, similar to the Extended Kalman Filter, will also be implemented.

REFERENCES

[1] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[2] D. Crisan and B. Rozovskii, Eds., *The Oxford Handbook of Nonlinear Filtering*. Oxford University Press, 2011.

[3] N. N. Taleb, *The Black Swan: The Impact of the Highly Improbable*, 1st ed. London: Random House, 2008.

[4] M. Idan and J. L. Speyer, "Multivariate Cauchy estimator with scalar measurement and process noises," *SIAM Journal on Control and Optimization*, vol. 52, no. 2, pp. 1108–1141, 2014.

[5] A. Y. Aravkin, J. V. Burke, and G. Pillonetto, "Robust and trend-following student's t kalman smoothers," *SIAM Journal on Control and Optimization*, vol. 52, no. 5, pp. 2891–2916, 2014.

[6] J. Fernández, J. L. Speyer, and M. Idan, "Stochastic estimation for two-state linear dynamic systems with additive Cauchy noises," *IEEE Transactions on Automatic Control*, vol. 60, no. 12, 2015.

[7] M. Rada and M. Černý, "A new algorithm for enumeration of cells of hyperplane arrangements and a comparison with Avis and Fukuda's reverse search," *SIAM Journal on Discrete Mathematics*, vol. 32, no. 1, pp. 455–473, 2018.

[8] D. Avis and K. Fukuda, "Reverse search for enumeration," *Discrete Applied Mathematics*, vol. 65, no. 1, pp. 21–46, 1996, First International Colloquium on Graphs and Optimization.

[9] S. Boyd and L. Vandenberghe, *Convex Optimization*. USA: Cambridge University Press, 2004.

[10] N. Duong, M. Idan, R. Pinchasi, and J. Speyer, "A note on hyper-plane arrangements in $\mathbb{R}^d$," *Discrete Mathematics Letters*, vol. 7, pp. 79–85, July 2021.

[11] N. Ploskas and N. Samaras, "Efficient GPU-based implementations of simplex type algorithms," *Applied Mathematics and Computation*, vol. 250, pp. 552–570, 2015.

[12] A. Gurung and R. Ray, "Simultaneous solving of batched linear programs on a GPU," in *ICPE '19: Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering*. Association for Computing Machinery, 2019.

[13] J. L. Speyer, M. Idan, and J. Fernández, "The two-state estimator for linear systems with additive measurement and process cauchy noise," in *51st IEEE Conference on Decision and Control (CDC)*, 2012, pp. 4107–4114.