VIVA: AN ONLINE ALGORITHM FOR PIECEWISE CURVE ESTIMATION USING ℓ^0 NORM REGULARIZATION

Richard B. Voigt

A DISSERTATION

in

Electrical and Systems Engineering

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

2015

Supervisor of Dissertation

Jonathan M. Smith
Olga and Alberico Pompa Professor of Engineering and Applied Science

Graduate Group Chairwoman

Saswati Sarkar
Professor, Electrical and Systems Engineering

Dissertation Committee
Ali Jadbabaie, Alfred Fitler Moore Professor of Network Science, Electrical and Systems Engineering
Jonathan M. Smith, Olga and Alberico Pompa Professor of Engineering and Applied Science, Computer and Information Science
Saleem Kassam, Solomon and Sylvia Charp Professor, Electrical and Systems Engineering
Insup Lee, Cecilia Fitler Moore Professor, Computer and Information Science
George Kramer, Professor, Anesthesiology, University of Texas Medical Branch

VIVA: AN ONLINE ALGORITHM FOR PIECEWISE CURVE ESTIMATION USING ℓ^0 NORM REGULARIZATION

Acknowledgements

Doctoral work is not performed in a vacuum (after all, even the mighty doctoral student is not immune to asphyxiation), and mine has been no exception to that rule. I count many blessings bestowed by the providential hand of a great God and His Son, the Lord Jesus Christ – loving family and friends, innate abilities and the professors and colleagues in university and industry who developed them, opportunities to use them – these have been no less essential to my journey than breath itself. To name each individual who has guided, advised, encouraged, or otherwise assisted me would eclipse any dissertation... and my memory is not up to the task in any case. Nevertheless, I would like to offer special thanks to several who stand out.

From the very beginning I have been influenced by my mother Dr. Mary Voigt (PhD), who taught me mathematics, reading, scientific methods, to always look for ulterior motives, and that skepticism and understanding one's relationship to God are not mutually exclusive and my father Richard H. Voigt, who has never let me forget that honorifics and intelligence are worth nothing without honor and integrity. I thank them for their unceasing love and all the ways they've showed it. And if my grandparents and siblings have shown me less love, it is only in degree and not in kind.

I thank my elementary and high school teachers for teaching me, as much about the expectations of society as about the course topics, and particularly for a lesson in Pre-Calculus about using extraordinary gifts to help those around me instead of alienating them (although it took me a few more years to achieve passing competence). KSU EECE professors William Kuhn, PhD and Don Gruenbacher, PhD, thank you for a first taste of engineering research, via poster presentations hung on the walls of Rathbone Hall, and thanks to James DeVault, PhD and Andrew Rys, PhD who convinced me to apply for graduate fellowships so I could perform research myself. Also to Mr. George Beck and KSU professors Medhat Morcos, PhD and Steve Warren, PhD for seeing more potential in me than I could in myself... and whose recommendations swayed the NDSEG fellowship and Penn admissions committees to agree. NASA flight surgeon Dr. Douglas Hamilton, MD I thank for getting me interested in medical data and devices and teaching me everything I know about cardiology (and Canada), in addition to no small amount of personal mentoring. Drs. Don Hummels (PhD) and Russ Meier (PhD) I credit with focusing me on practical engineering and real-time signal processing, and then Penn ESE professors Ali Jadbabaie, PhD, George Pappas, PhD, and Srisankar Kunniyur, PhD took those tools to the next level by adding a theoretic framework. Also Dr. Santosh Venkatesh (PhD), who installed a healthy fear and awe of mathematics that most of my peers had acquired much earlier, and whom I will never, ever bet against.

In addition, I would like to thank the administrators and staff of KSU Engineering and the EECE department, Penn Engineering and the ESE department, and the UTMB Anesthesiology department. Most probably these brave souls are responsible for making the sun rise each morning; certainly they keep the doors open and the wheels turning smoothly, and have been of particular benefit to my forgetful self. In addition, my friends and church family kept me from

sinking into despair as I dealt with the disintegration of a particularly important personal relationship, and helped me prioritize. Coworkers at Wyle, UTMB, and Sparx helped me maintain balance while juggling work, school, church, and personal issues, backfilled for me when I needed a more intense focus on academic work, and most recently courageously served, without pay, as proofreaders.

Dr. George Kramer (PhD), professor of anesthesiology and director of the Resuscitation Research Lab at UTMB, thank you for correcting all my prior misconceptions about what research is (or at least, replacing them with your own), welcoming me into a medical research environment and freely sharing project ideas and data, supporting conference travel and poster presentations, introducing me to world-famous scientists, showing your dedication to personal leadership through sharing proverb after (often completely fabricated and usually inappropriate) anecdote, and always, ALWAYS being willing to buy me more beers than I was willing to drink.

To the other students of the Penn CIS Distributed Systems Laboratory, for fitting me into your presentation schedule, listening to several half-baked and wholly disorganized slide sets over the years, and always making me feel welcome, thank you. To all the researchers of the UTMB Resuscitation Research Lab, postdocs, technicians, nurses, and staff anesthesiologists alike, I am indebted for your efforts to put those half-baked ideas into practice in animal surgeries and collect the data I requested, for enduring my total misconceptions about physiology and carefully (and oh-so-patiently) setting me straight over many months, and for the friendships that grew.

I would also like to thank the LaTeX developers, the MiKTeX package maintainers, the tex.stackexchange.com community, and especially Dr. Christian Feuersänger for his pgfplots package, without which this document would be far less attractive. A gift of Visual Studio from the Microsoft MVP program has also been most helpful.

Finally, and most important, my advisor, dissertation supervisor, and friend, Dr. Jonathan Smith (PhD), who taught me the art of turning priorities into actions, explained the brotherhood of PhD engineers in terms even I could understand, gave invaluable feedback and direction on this entire undertaking and kept me on track across more than a decade despite my inadequacies and competing demands, deserves more thanks than I can express.

My successes I owe to all those named above as well as others who have labored without recognition; mistakes and omissions are my own.

I also acknowledge the following sources of funding which made possible my doctoral studies, research experiments which generated data for my use, and my analysis:

- National Defense Science and Engineering Graduate Fellowship, award years 2002-2005
- National Science Foundation, grant CNS-1040672 “FIA: NEBULA“
- Olga and Aberico Pompa Professorship, an endowed chair at the University of Pennsylvania
- Office of Naval Research, grant N00014-12-C-0556 “Decision Support and Closed Loop Control Systems”
- U.S. Army Medical Research and Materiel Command, Cooperative Agreement #09078006 “Titrated Bolus Resuscitation“
- National Institutes of Health, grant 1R01HL092253-01 “Decision-Assist and Closed-Loop Resuscitation of Burn-Injured Patients”
- Sparx Engineering

ABSTRACT

VIVA: AN ONLINE ALGORITHM FOR PIECEWISE CURVE ESTIMATION USING ℓ^0 NORM REGULARIZATION

Richard B. Voigt

Jonathan M. Smith

Many processes deal with piecewise input functions, which occur naturally as a result of digital commands, user interfaces requiring a confirmation action, or discrete-time sampling. Examples include the assembly of protein polymers and hourly adjustments to the infusion rate of IV fluids during treatment of burn victims. Estimation of the input is straightforward regression when the observer has access to the timing information. More work is needed if the input can change at unknown times. Successful recovery of the change timing is largely dependent on the choice of cost function minimized during parameter estimation.

Optimal estimation of a piecewise input will often proceed by minimization of a cost function which includes an estimation error term (most commonly mean square error) and the number (cardinality) of input changes (number of commands). Because the cardinality (ℓ^0 norm) is not convex, the ℓ^2 norm (quadratic smoothing) and ℓ^1 norm (total variation minimization) are often substituted because they permit the use of convex optimization algorithms. However, these penalize the magnitude of input changes and therefore bias the piecewise estimates. Another disadvantage is that global optimization methods must be run after the end of data collection.

One approach to unbiasing the piecewise parameter fits would include application of total variation minimization to recover timing, followed by piecewise parameter fitting. Another method is presented herein: a dynamic programming approach which iteratively develops populations of candidate estimates of increasing length, pruning those proven to be dominated. Because the usage of input data is entirely causal, the algorithm recovers timing and parameter values online. A functional definition of the algorithm, which is an extension of Viterbi decoding and integrates the pruning concept from branch-and-bound, is presented. Modifications are introduced to improve handling of non-uniform sampling, non-uniform confidence, and burst errors. Performance tests using synthesized data sets as well as volume data from a research system recording fluid infusions show five-fold (piecewise-constant data) and 20-fold (piecewise-linear data) reduction in error compared to total variation minimization, along with improved sparsity and reduced sensitivity to the regularization parameter. Algorithmic complexity and delay are also considered.

Contents

List of Tables

List of Figures

CHAPTER 1

Background

This chapter describes the medical scenario that motivated development of a novel online denoising algorithm. Shock, defined as "a life-threatening, generalized maldistribution of blood flow resulting in failure to deliver and/or utilize adequate amounts of oxygen" (Antonelli et al., 2007) occurs acutely in victims of trauma, infection, and burn. Resuscitation of shock requires individualized care that accounts for patient-specific responses, as responses that differ from the "textbook case" are prevalent. Computers can assist with provision of fluid and drug therapies used to resuscitate shock, as well as decision support and alarming. However, hemodynamic optimization requires accurate and timely measures of both therapy delivered and patient response. The harried environments in which medicine is practiced make obtaining these signals quite challenging. Existing methods of noise removal are poorly suited to the task.

1.1. The Value of Infusion Monitoring

1.1.1. Shock is a Major Factor in Preventable Deaths. Traumatic injuries cause more deaths of patients aged 40 and under than any other cause (Spinella and Holcomb, 2009), causing approximately 90,000 deaths annually in the USA alone. Of these, Spinella and Holcomb estimate that 10,000 or more result from inadequate treatment of hemorrhagic shock resulting from survivable injuries. Amongst battlefield injuries, the prevalence of traumatic hemorrhage is even higher, with one fifth of fatalities occurring following injuries classified as potentially survivable, due to exsanguination before reaching a medical treatment facility (Eastridge et al., 2012). According to Newgard et al. (2010), receiving care from an experienced trauma team at a major trauma center is crucially important to outcome, while time spent pre-hospital and in transport is not, but this considered only transport times up to two hours and assumes that adequate resources are available to keep the casualty stable, something that often is not true on the battlefield.

The US military has proposed to address this need by (1) using electronics to bring expertise to combat medics (2) evacuate casualties to definitive care earlier (3) improve care during transport . Several key initiatives include use of unpiloted drones, having delivered supplies to forward troops, to perform timely medical evacuation through environments too dangerous to hazard human care providers (ONR BAA 12-004 CONOPs). An autonomous critical care system (ACCS) is under development by the Office of Naval Research. The ACCS will provide decision support to a corpsman in the field to help stabilize and prepare a patient for transport, then manage a patient during medical evacuations, either autonomously or with the help of remote guidance from telemedicine caregivers (ONR BAA 11-012). The ACCS can also be used in medical facilities as a force multiplier by aiding in hemodynamic management, thus requiring less medical provider time per patient.

Shock resulting from severe sepsis has a mortality rate from 28 % to 50 % (Wood and Angus, 2004), killing over 100,000 hospital patients annually (Martin et al., 2003). Burn patients also are at risk of hypotensive shock, either from fluid loss induced by the burn, or resulting from secondary infections.

The need for improvements in treatment and prevention of shock is clear.

1.1.2. Shock is Treated by Optimum Fluid Therapy. Infusion of fluid to increase circulating blood volume (Sanford and Herndon, 2001) is essential to resuscitation of shock victims because it sequentially increases venous return, cardiac stroke volume, and volumetric flow rate ("cardiac output") (Gagnon, 2009; Antonelli et al., 2007; Kramer et al., 2007a). Increased cardiac output will, ceteris paribus, both increase the net oxygen delivered to tissues and increase tissue perfusion. However, fluid infusion is a two-edged sword, as fluid overload also puts the patient at risk (Perel, 2008; Spinella and Holcomb, 2009). Increased vascular pressure increases effluent flow rates, making it more difficult for clots to form (coagulation), and also increases strain on newly formed clots, which may cause them to fail and internal bleeding to resume (Spinella and Holcomb, 2009). Dilution of blood with infusate also reduces the concentration of oxygen carriers and clotting factors (Spinella and Holcomb, 2009).

Additional effects are related to characteristics of the particular fluid used (Guha et al., 1996; Kramer et al., 2007a). Use of saline solutions causes a decrease in protein concentrations, reduced colloid osmotic pressure, and increased leakage of water into the surrounding tissues (Saffle, 2007; Kramer et al., 2007a), causing swelling ("edema"), while solutions of hypertonic saline or colloids have the opposite effect Guha et al. (1996); Fodor et al. (2006). If this occurs in the lungs ("pulmonary edema"), the increased aveolar wall thickness will reduce the effectiveness of gas exchange (Holm et al., 2004), and may result in lower arterial oxygen saturation. If swelling occurs inside the abdomen, pressure may be exerted on vital abdominal organs, causing collapse of veins and impeding blood flow and organ function. This condition is known as "intra-abdominal hypertension" and if unchecked leads to organ failure designated "abdominal compartment syndrome" (Saggi et al., 2001; Pham et al., 2008; Salinas et al., 2008; Cancio, 2014). Edema also increases the risk that a burn injury will become septic (Hoskins et al., 2006; Salinas et al., 2008); conversely sepsis increases edema (van der Heijden et al., 2009). If the infusate is not warmed to body temperature, it can contribute to hypothermia, negatively affecting coagulation that may already be compromised by acidosis (Tsuei and Kearney, 2004).

For all these reasons it is desirable to infuse the minimal volume needed to achieve clinical goals (Cancio, 2014). Yet providers continue to infuse burn patients with significantly more fluid (Kramer et al., 2007b; Salinas et al., 2008; Oda et al., 2006) than called for by consensus recommendations (Pham et al., 2008), and these large fluid volumes are implicated in development of abdominal compartment syndrome (ACS) (Oda et al., 2006; Saffle, 2007). Resuscitation fluid is associated with ACS in non-burn patients as well (Balogh, 2003; Sugrue, 2005). According to Sugrue (2005), 5 % of intensive care patients suffer from ACS. Mortality varies between 25 % and 75 % (Balogh et al., 2003).

1.1.3. Goal-Directed Therapy Achieves Better Outcomes. One possible explanation for the persistence of burn care providers in choosing to err on the side of extra fluid is that under-resuscitation represents acute risk, while harm resulting from over-resuscitation manifests multiple hours if not days in the future. Often the providers are convinced that fluid volumes exceeding the formulaic calculations are truly needed in these patients. Yet evidence shows that when fluid is systematically managed using objective goal-directed rules, fluid volumes are generally less than predicted by the Parkland formula without leading to under-resuscitation (Salinas et al., 2011; Oda et al., 2006; Arlati et al., 2007), and in fact achieve better outcomes (Salinas et al., 2008).

The advantages of goal-directed (explicit feedback loop) therapy are not limited to burn patients either. Outcomes in sepsis patients improved following implementation of goal-directed therapy (Otero, 2006; Perel, 2008). A meta-analysis performed by Hamilton et al. (2011) found significant improved outcomes in a broad range of high risk surgical and critical care patients.

In addition, adherence to clinical guidelines based on published evidence and expert consensus leads to better and more consistent outcomes than achieved by care providers making decisions individually (Fakhry et al., 2004; Levy et al., 2010; Morris et al., 2011; Krinsley, 2004).

Fortunately, both knowledge-based treatment and feedback control are amenable to computer automation.

1.1.4. Research Demonstrates that Decision Support and Closed-Loop Control of Hemodynamics is Possible. In fact, there have already been successes in implementation of closed-loop management of fluid (Rafie et al., 2004; Ying et al., 2002; Hoskins et al., 2006; Salinas et al., 2008; Kramer et al., 2008; Vaid et al., 2006; Rinehart et al., 2011, 2012; Meador, 2014; Cancio, 2014), vasodilators for relief of hypertension (Hammond et al., 1979; Ying et al., 1992), and vasoconstrictors for correction of normovolemic hypotension (Yu et al., 1992; Rao et al., 1999, 2003; Ngan Kee et al., 2008) in large mammals and human patients. Automatic sedation has been demonstrated as well (Bibian et al., 2005; Hemmerling, 2009; Struys et al., 2001). These closed-loop control studies have been performed during a variety of surgical and intensive care scenarios as well as first responder / point of injury simulations. Even though the computer does not have knowledge of upcoming surgical actions and consequences, the responsiveness of silicon processors results in making decisions about new data more quickly; reaching targets faster, with less overshoot; and managing endpoint variables more precisely than can a human provider with multiple responsibilities.

1.1.5. Limitations of Automated Fluid Delivery using a Fixed Control Law. While the successful closed-loop tests demonstrate the potential rewards of using computers to automatically manage hemodynamics, these studies are limited in scope and it would be premature to believe that the algorithms are ready for broad deployment. In particular, the research animals were healthy apart from the intentionally inflicted injury and patients were selected using criteria that excluded complications. While these restrictions are perfectly understandable considering the research goals of reproducibility and cohort comparison, and the ethical goal of patient safety, they do not address questions about safety and efficacy on a broader population. Inter-patient variability that was largely excluded by study criteria may lead to poor outcomes. For example, controlling fluid infusion to sepsis patients based on a target of central venous pressure may cause overinfusion (Perel, 2008). According to Preisman (2005), single-variable optimization of cardiac output may lead to fluid overload; as many as 50 % of patients may be unresponsive (in the sense of increasing stroke volume) to fluid.

In extreme cases, the negative effects of fluid infusion could adversely affect the feedback variable, setting up a positive feedback loop that demands fluid therapy more insistently even as it drives the patient ever further from the target. In their work on closed-loop sedation, Bibian et al. (2005, 2004); Zikov and Bibian (2014) explain the importance of bounding the uncertainty of patient response and defining the controller's region of convergence and stability. A key insight comes from Haddad and Bailey (2009), "[...] it has been assumed that stability follows from the pharmacokinetic/pharmacodynamic model. However, this is not the case since these controllers do not account for full model uncertainty, unmodeled dynamics, exogenous disturbances, and system nonlinearities."

1.1.6. Controllers Benefit from Automated Response Analysis. Individualization of models through observing output response for the purpose of per-patient parameter identification, even if only a subset of parameters can be determined (partial parameter identification), can be used to improve controller performance (Bibian et al., 2004). Some control techniques, such as model adaptive control, explicitly use the subject-specific parameter values. Other controllers may be able to account for reduced uncertainty by gain adjustment.

According to Haddad and Bailey (2009), robust controllers sacrifice performance for stability, while adaptive control seeks both performance and stability. This presents a design tradeoff, since both present risks: Overdamping used to increase controller robustness causes slow convergence and increases the time a patient is left in a dangerous physiologic state. Additionally the overdamped control loop is less capable of opposing the effect of large disturbances. On the other hand, adaptive control is heavily reliant on the fidelity of parameter estimates, and can become unstable if those estimates become corrupted.

1.1.7. "Electronic Doctor" Diagnosis and Prescription will Rely on Automated Response Analysis. Like the human decision process (Kahneman, 2011), autonomous critical care systems of the future will consist of a slow "reasoning" expert system performing diagnosis and selection of therapies and drugs, with fast "reacting" closed-loop controllers carrying out these therapies. Several closed-loop controllers have already been mentioned. Gholami et al. (2012) states that

> It is important to note that expert systems are already in widespread use in other branches of medicine, more prominently in disease diagnosis, where the system inputs are the patient's details and symptoms, and the system outputs are probable diagnoses, recommended treatments or drugs which may be prescribed. Such systems are typically open-loop and may be regarded as rule-based search engines to help the clinician in his/her mapping of a given set of symptoms to a possible cause (disease).

and furthermore provides a framework for application of Bayesian expert systems in the fast closed-loop portion of the system. This is typically seen as initial encoding of care provider knowledge heuristics, which are later replaced by robust and adaptive controllers as dynamical system models are developed.

One key element that remains is the forwarding of performance information from the reaction layer to the reasoning layer, which may take actions such as alarming, swapping controller implementations for one expected to yield better performance on the particular patient, or discontinuation of an ineffective therapy. Parameter estimation using Kalman filters has been described in Luspay and Grigoriadis (2014). This and similar adaptive techniques will provide the reasoning system with the information needed to manage inter-patient variability.

1.2. Practical Measurement of Infusion Rate

1.2.1. Effects of Erroneous Infusion Rate. Because assessment of response to fluid bolus is intended for computer-assisted diagnosis, computerized alarming, closed-loop controllers checking preconditions that assure controllability and stability, and eventually for automated planning of patient care, the hazard of basing this assessment on invalid input data has potential to lead to harm. The assessment aggregates infusion and response data over a time interval, providing reduced sensitivity to small or transient errors in the input signal, so the input signal (infusion rate) must be known within these tolerances.

1.2.2. Need for Independent Rate Measurement. When the infusion rates of IV pumps are set by computerized control, it is tempting to accept the control signal as a perfectly accurate input signal. However, actual flow rates vary from requested rates due to variations in resistance, source pressure, and backpressure, as well as automatic safety shutdown if air or occlusion is detected. Hospital pumps controlled via front panel may or may not report the currently set rate via a computer interface. Moreover, care providers give some fluids without the aid of pumps. Manually delivered fluids are particularly common in operating rooms. Furthermore, some infusers offer relative control of flow without providing accurate flow rate information. Controllers which fail to account for actual flow rates lead to windup and improper adaptation (Haddad and Bailey, 2009). For these reasons, having an independent measure of infusion rate is helpful.

In a research laboratory, flow rates can be accurately and continuously measured using electronic flow probes (e.g. transit time or Doppler-derived rate). But these devices are expensive and require exact calibration and careful handling to protect the transducer crystals, making them unsuitable for routine use. A simple, lightweight, and inexpensive method for monitoring infusions is to weigh the IV bag or other source reservoir using a load cell. Effectiveness for response assessment depends on the accuracy of flow rate estimates derived from the load cell

signals. Environments with significant mechanical vibration pose particular obstacles to using load cells for infusion monitoring. Even in fixed settings, small errors in weight measurements lead to large uncertainty in rate calculations. For example, an error of only 0.1 g, if the time resolution is 10 seconds, leads to a flow rate error of 0.6 mL/min or 36 mL/h. And the problem increases with higher time resolution.

1.2.3. Infusion Monitoring System Prototype. A four channel "smart IV pole" IV bag monitoring system to use load cells to continuously measure the weight of IV bags has been conceived in the UTMB Resuscitation Research Laboratory (Galveston, TX) and designed and built by Sparx Engineering (Manvel, TX). UTMB researchers developing smart hemodynamic resuscitation components for the Navy ACCS have used this load cell system to record infusion data during fluid resuscitation studies of hemorrhagic and burn shock in animal models. The Texas Instruments LMP90099 analog-to-digital converter used in this system samples load cell readings 53.66 times per second with a system error of ±1 g (0.2 g after averaging). The system undergoes two-point offset and gain calibration before each use. Data collected using this system consists of periods of valid readings with additive limited measurement noise, interspersed with short-duration burst noise caused by mechanical forces on either side of the load cell (IV pole or IV bag). In addition, following manipulation of the IV bag, the data are frequently observed to exhibit underdamped oscillation due to pendulum action of the IV bag.

Several different approaches may be taken to denoising the flow rate estimate. Classical lowpass filtering trades time resolution for reduced error. Regularization methods rooted in optimization theory attempt to achieve low errors and good time resolution by finding the simplest signal that is consistent with the measured data. A regularization problem was stated which encodes the domain knowledge that care providers infrequently change the infusion rate. I developed the Viterbi-Inspired Variation Assessment (VIVA) method to efficiently perform this ℓ^0 norm regularization. Further development of the VIVA method added time-varying weighting to reduce sensitivity to burst noise.

Denoising is less critical for other applications of the load cell data, such as detection of IV bag depletion or calculation of cumulative delivered fluid, which are impacted less by measurement noise. But these too benefit from identification of bag changes and rejection of burst noise from mechanical impacts on the IV pole.

1.3. Existing Methods for Denoising Piecewise Signals

1.3.1. Bicriterion Model

In the search for the maximum-likelihood estimate $\{\tilde{x}_k\}$ of the underlying signal $\{x_k\}$, it is natural to use a bicriterion objective function which seeks both agreement with the observations $\{y_k\}$ and simplicity of description. Simplicity may be motivated by domain knowledge that control actions are few or because a short (compressed) representation is desired to simplify communication, storage, and further analysis. Then

$$\tilde{\mathbf{x}} = \operatorname*{argmin}_{\mathbf{x}'} \epsilon(\mathbf{x}') + \zeta f(\mathbf{x}') \tag{1.1}$$

where $\epsilon(\mathbf{x}')$ represents a goodness of fit "distance" from the observation sequence, $f(\mathbf{x}')$ represents the control burden, and ζ is a weighting coefficient controlling a tradeoff between the two.

The usual metric for goodness of fit is integral-square residual error, which for sampled measurements becomes a discrete sum:

$$\epsilon(\mathbf{x}') = \sum_{k=1}^{N} (x'_k - y_k)^2 \tag{1.2}$$

This corresponds to negative log-likelihood in the case of additive i.i.d. Gaussian noise (AWGN). In the case of non-stationary noise, a weighting function may be included:

$$\epsilon(\mathbf{x}') = \sum_{k=1}^{N} c_k \left(x'_k - y_k\right)^2 \tag{1.3}$$

For control actions which act on either value or rate of observations, the control burden may be expressed as the number (cardinality) of non-zero control inputs

$$f(\mathbf{x}') = \operatorname{card}\left(\Delta^p \mathbf{x}'\right) \tag{1.4}$$

where p expresses the controller-observer relationship, with particular interest in the values

$$p \in \begin{cases} 2 & \text{rate control} \\ 1 & \text{value control} \end{cases} \tag{1.5}$$

This problem is relevant to many areas of operations research, with related work appearing in the changepoint, linear programming, integer programming, dynamic programming, and convex optimization literature. Existing work falls into several classifications:

- In-exact solutions, typified by greedy algorithms
- Convex relaxations
- Exact solutions with poor scaling
- Dynamic programming

1.3.2. Inexact Methods

1.3.2.1. "Greedy" Step-Fitting Algorithms. Another active research area in biology which requires denoising piecewise-constant data is the study of protein regulation of microtubule assembly (Kerssemakers et al., 2006). In 2008, Carter et al. used artificial data sets analogous to kinesin motion to compare four methods for automatic resolution of individual steps, concluding that the best of these was the chi-squared minimization method developed by Kerssemakers et al. (2006). This method uses a fractal-like procedure whereby the entire dataset is split into two at the point which minimizes residual chi-square measure. The procedure is then repeated on each segment found, and the new step which reduces residual chi-square the most is accepted. The algorithm also provides a quality measure which terminates the iteration. In the developers' own words, "Once found, the step-locations do not change anymore, although the associated step sizes continue to change as they depend on the location of the neighboring steps" making this a greedy algorithm. This approach is similar to the Binary Segmentation method described in the changepoint literature (Killick et al., 2012), but the Kerssemakers et al. method incorporates an additional termination condition designed to prevent overfitting.

1.3.3. Convex Methods

1.3.3.1. Total Variation Minimization. Little and Jones sought to overcome the limitations of the greedy step-fitting algorithms using a global ℓ^1 regularization solved using convex optimization techniques. They demonstrated that their lasso filter based on total variation minimization yields results with lower noise for synthesized piecewise constant data than classical median filters used to preprocess data for the greedy step-fitting algorithms. Their algorithm also performed well on bacterial motor rotation data. The recent development of Little's approach and its superior performance compared to the methods in Carter et al. (2008) establishes it as the baseline to beat for piecewise-constant models. However, the total variation minimization approach yields two types of systematic error. As seen in Figure 1, segments which lie to the same side of both neighbors are systematically estimated with a bias toward the neighboring levels, underestimating step magnitude. Furthermore, segments which lie between neighboring levels carry no penalty at all, resulting in stairstep output (Figure 2).

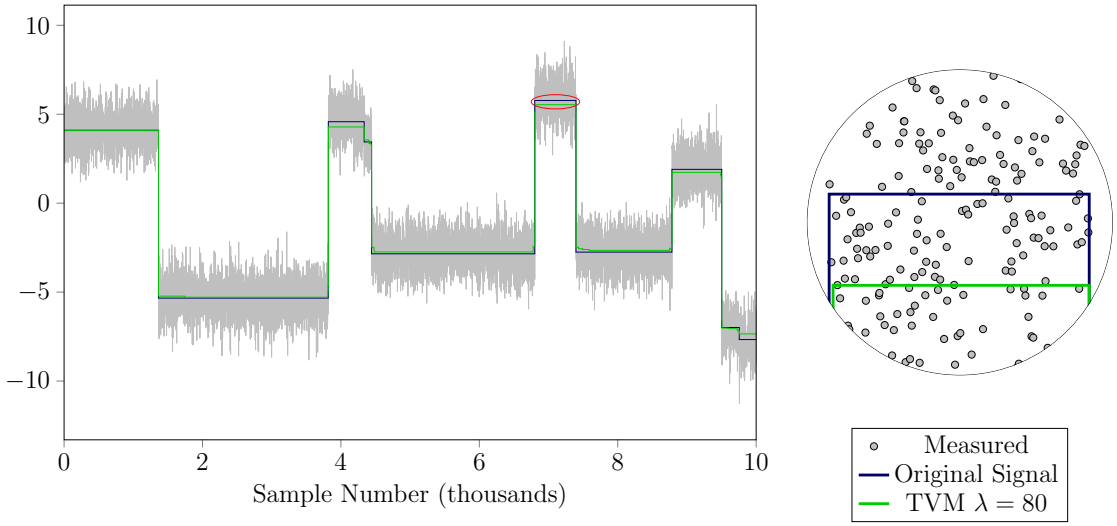


FIGURE 1. In Total Variation Minimization, the penalty function biases the estimate for extreme segments

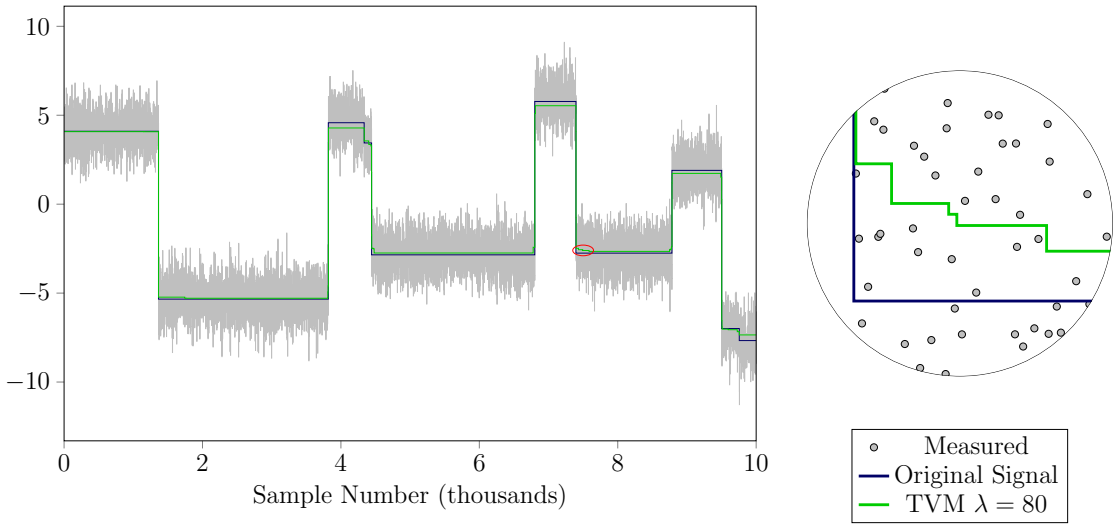


FIGURE 2. Stairsteps form in Total Variation Minimization, because the penalty function assigns zero incremental cost to segments which lie between neighbors

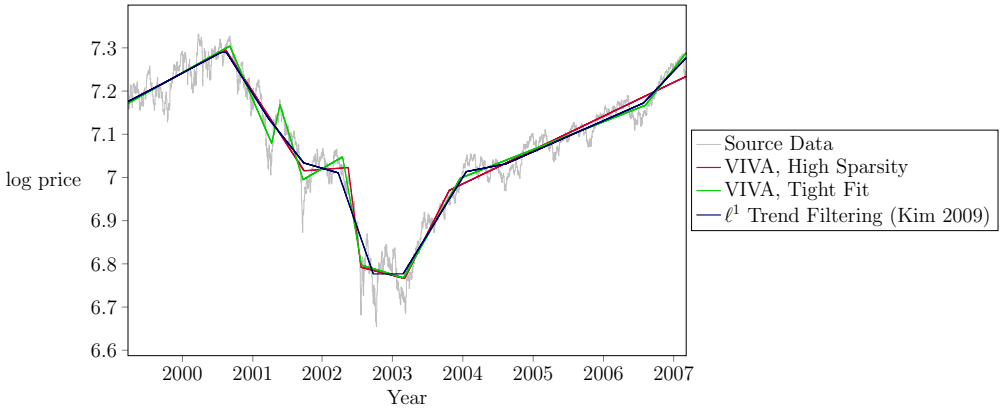


FIGURE 3. ℓ^1 Trend Filter is systematically biased toward shallow changes in slope

TABLE 1. Comparison of residual error and sparsity of trend-filter estimates

Method	RMS residual	Segmentation	
		# Changepoints	Slope Threshold
ℓ^1 Trend Filtering	0.0314	12	10^{-4}
(Kim et al., 2009)		42	10^{-6}
VIVA (Tight Fit)	0.0283	9	10^{-10}
VIVA (Sparse)	0.0319	6	10^{-10}

1.3.3.2. Trend Filtering. Kim et al. demonstrated use of global ℓ^1 bicriterion regularization for fitting polyline models. These models are used extensively in a variety of financial and biological applications (Kim et al., 2009). Considering the stock price curve fitting problem which appeared as Figure 2 in Kim et al. (2009), comparison of the ℓ^1 trend filtering result to two other piecewise-linear curves shows that this relaxation exhibits the same systematic sub-optimality as total variation minimization, manifesting as underestimation of changes in slope and failure to obtain a truly sparse description (Figure 3, Table 1).

1.3.3.3. Other Interior Point Methods. Julian et al. (1998) performs fitting of piecewise-affine multidimensional surfaces to sample data by proposing partition boundaries and performing descent using a Newton-Gauss algorithm to exponentially converge to a local minimum. When the conditions for global optimality are not met, the algorithm attempts to escape local minima by restarting the algorithm from new initial partitions. Problems with larger numbers of hyperplanes are addressed by optimizing using a smaller number to obtain an initial partitioning for the larger problem. This method is avoided for data with large numbers of segments due to its poor scaling with facet count.

Xu et al. (2011) fits images with a reduced variation cardinality using an iterative algorithm that alternates between identifying edges and then smoothing between them. Their method performs simultaneous optimization across the entire dataset and works only for the piecewise-constant case. Since less expensive methods exist for finding exact solutions in the one dimensional case, no attempt will be made to adapt it for real-time use.

1.3.3.4. Advantages of ℓ^0 Regularization. It is interesting that, aside from its convexity, the most valuable attribute of ℓ^1 norm regularization is that it approximates finding solutions to the ℓ^0 norm regularization problem (Boyd and Vandenberghe, 2004; Ramirez et al., 2013; Kim et al., 2009), or even exactly finds sparse solutions (Donoho, 2006). Chartrand (2007) uses a ℓ^p norm with $p < 1$, in order to get closer to ℓ^0, making the observation "The resulting optimization

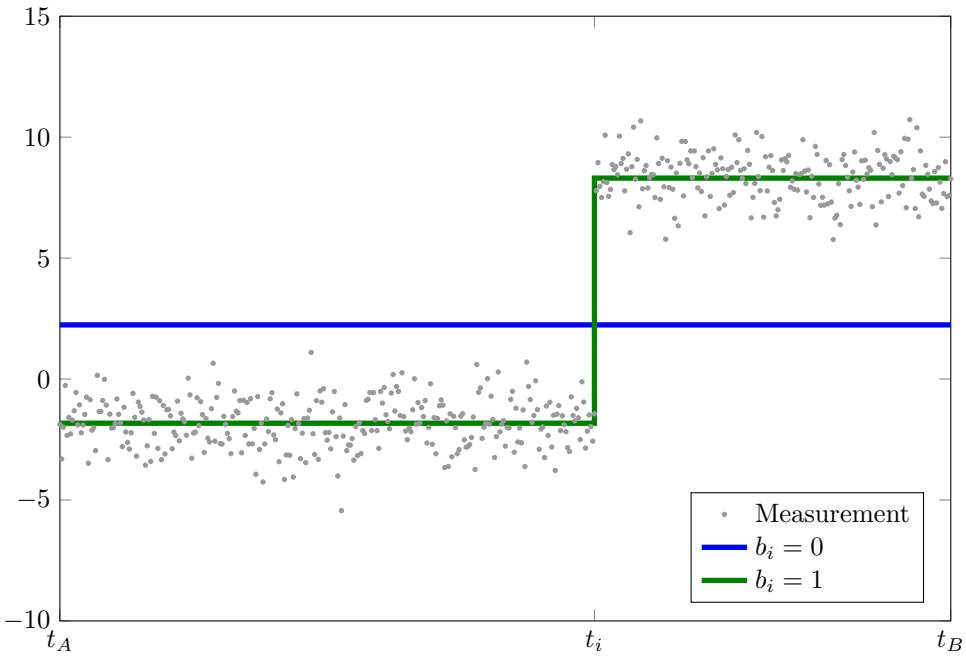


FIGURE 4. The binary sequence b_j segments the measurement sequence y_j into independent constant segments

problem will not be convex, and is described in the literature as intractable." Undissuaded, Chartrand then goes on to show that using descent methods with this non-convex ℓ^p norm is statistically likely to find a solution with the globally optimal sparsity pattern, and that even local minima generate better reconstructions of sparsely sampled signals than found using the ℓ^1 norm. Furthermore Xu et al. (2011) constructs signals for which the ℓ^1 norm heuristic fails to find the optimal sparse solution.

1.3.4. Equivalent Mixed-Integer Models

Another expression of the maximum likelihood estimation problem is as a mixed-integer quadratic program (Roll et al., 2004). Augment the problem with additional binary variables, which segment the measurement sequence (Figure 4):

$$b_k = \begin{cases} 1 & \exists i, (k-1)T_v < \mathcal{T}_i - t_1 \leq kT_v \\ 0 & \text{otherwise} \end{cases} \tag{1.6}$$

where T_v is the time resolution with which it is desired to locate step changes.

The complete MIQP model for piecewise-constant estimation with the ℓ^0 norm (variation cardinality minimization) objective function is:

$$\begin{array}{lrcccl} \text{minimize} & (\mathbf{x}' - \mathbf{y})^T (\mathbf{x}' - \mathbf{y}) + \zeta \mathbf{1}^T \mathbf{b}' & & & & \\ \text{subject to} & -x'_k + x'_{k+1} - Mb'_k & \leq & 0 & & \forall k \\ & x'_k - x'_{k+1} - Mb'_k & \leq & 0 & & \forall k \\ & b'_k \in \{0, 1\} & & & & \forall k \end{array} \tag{1.7}$$

The piecewise-linear estimate with ℓ^0 norm objective function and continuity constraints has a similar canonical MIQP form:

(1.8)

$$\begin{array}{lccccccc} \text{minimize} & & & (\mathbf{x}'-\mathbf{y})^T(\mathbf{x}'-\mathbf{y})+\zeta\mathbf{1}^T\mathbf{b}' & & & & \\ \text{subject to} & (t_i-t_{i+1})x'_{k-1} & +(t_{i+1}-t_{i-1})x'_k & +(t_{i-1}-t_i)x'_{k+1} & -Mb'_k & \leq & 0 & \forall k \\ & (t_{i+1}-t_i)x'_{k-1} & +(t_{i-1}-t_{i+1})x'_k & +(t_i-t_{i-1})x'_{k+1} & -Mb'_k & \leq & 0 & \forall k \\ & & & b'_k \in \{0,1\} & & & & \forall k \end{array}$$

So too is this piecewise-linear estimation problem with the continuity constraints removed (periodic sampling assumed for simplicity of notation):

$$\begin{array}{lccccccc} \text{minimize} & & & (\mathbf{x}'-\mathbf{y})^T(\mathbf{x}'-\mathbf{y})+\zeta\mathbf{1}^T\mathbf{b}' & & & & \\ \text{subject to} & -x'_{k-1} & +2x'_k & -x'_{k+1} & -Mb'_{k-1} & -Mb'_k & \leq & 0 & \forall k \\ & x'_{k-1} & -2x'_k & +x'_{k+1} & -Mb'_{k-1} & -Mb'_k & \leq & 0 & \forall k \\ & & & b'_k \in \{0,1\} & & & & & \forall k \end{array} \tag{1.9}$$

Although it is easier to solve exactly, the model is more complex.

In contrast, the ℓ^1 norm (total variation minimization) formulation is (piecewise-constant, mutatis mutandis for piecewise-linear with continuity):

$$\begin{array}{lccccc} \text{minimize} & & (\mathbf{x}'-\mathbf{y})^T(\mathbf{x}'-\mathbf{y})+\zeta\mathbf{1}^T\mathbf{d}' & & & \\ \text{subject to} & -x'_k & +x'_{k+1} & -d'_k & \leq & 0 \quad \forall k \\ & x'_k & -x'_{k+1} & -d'_k & \leq & 0 \quad \forall k \end{array} \tag{1.10}$$

which is a quadratic program (with linear constraints).

If instead of minimizing the integral-square residual error, the integral absolute residual error (ℓ^1 norm) is minimized,

$$\epsilon(\mathbf{x}') = \sum_{k=1}^{N} |x'_k - y_k| \tag{1.11}$$

then the objective function becomes linear, forming a mixed-integer linear program (MILP):

$$\begin{array}{lccccccc} \text{minimize} & & & \mathbf{1}^T\mathbf{r}'+\zeta\mathbf{1}^T\mathbf{b}' & & & & \\ \text{subject to} & -x'_k & +x'_{k+1} & & -Mb'_k & \leq & 0 & \forall k \\ & x'_k & -x'_{k+1} & & -Mb'_k & \leq & 0 & \forall k \\ & -x'_k & & +y'_k & -r'_k & \leq & 0 & \forall k \\ & x'_k & & -y'_k & -r'_k & \leq & 0 & \forall k \\ & & & b'_k \in \{0,1\} & & & & \forall k \end{array} \tag{1.12}$$

Naturally it is also possible to use the ℓ^1 norm for both goodness of fit and simplicity of description, in which case a linear program is obtained. Another related linear program is obtained from the minimax residual:

$$\epsilon(\mathbf{x}') = \sum_i \max_{\mathcal{T}_i \leq kT_v \leq \mathcal{T}_{i+1}} |x'_k - y_k| \tag{1.13}$$

1.3.5. Methods of Solving Integer Programs

Branch-and-bound is a standard technique for solving mixed integer programs: linear and quadratic programs with integer or binary variables. However, the presence of Big-M in constraints leads to relaxations with infeasible optimal solutions and weak bounds (Bertsimas and Shioda, 2007; Toriello and Vielma, 2012). In some cases Big-M can be replaced by a finite value, for example in piecewise-constant curve fitting, no step will ever be greater than the range of the observations. But these bicriterion regularization problems still produce weak relaxations as a rule, leading to excessive branching.

1.3.5.1. Symmetry reduction. Symmetry (when mapping between descriptive parameters and solution is not injective, for example when permutations of the parameters are equally valid) poses a particular problem for branch-and-bound, since even the optimal cost is not strong enough to prune subproblems that contain another optimum (Barnhart et al., 1998).

The binary variables are paired with observations in the MIQP models given, so if the observation times are distinct, there is no symmetry. In case of multiple observations at a single time, this could become problematic. Especially when there is no continuity constraint, if observations occur simultaneously with a jump discontinuity (to within measurement time precision), the wrong partition of observed values could lead to suboptimality. However rather than introducing symmetry, one need only consider two orderings, corresponding to either ascending or descending sort of the observations.

A similar approach is used by Roll et al. (2004) to address estimation of a monotonic piecewise-affine function (jump discontinuities allowed, provided they are in the direction preserving monotonicity) serving as the non-linear output mapping of a Wiener model. Roll et al., not having knowledge of the abscissa, breaks symmetry by sorting observations by the ordinate, resulting in computational complexity which is combinatorial – polynomial in the length of the input data, but exponential in the number of changepoints. This ordering could prove problematic if the effect of noise swaps two (or more) observations which lie either side of a changepoint. Due to this concern and the high complexity, this approach would not be used for the simpler case of finding piecewise functions of time.

1.3.5.2. Bound strengthening methods. Because branching increases complexity exponentially, effort is often expended on further constraining the linear / convex relaxation to produce a feasible solution. Branch-and-cut and branch-and-price (Barnhart et al., 1998) are complementary methods that tighten the bounds of the relaxed problem. Gomory cuts specifically tighten subproblems by adding bounds on a variable whose integrality is violated by the relaxed optimum. For MILP problems (linear objective function), various cutting planes including Gomory cuts, lifted-cover inequalities, flow cover inequalities, and gub-cover inequalities can be automatically added by such solvers as bc-opt (Cordier et al., 1999), MPSARX, MINTO, and MIPO. While not directly applicable to the MIQP models, it is likely that commercial solvers such as CPLEX (IBM) and would be able to apply cuts to reduce the required branching.

Branch-and-price aids in the optimization of problems with many variables (as these MIQP models have) by optimizing using a small basis set. Other integer variables are exchanged with the basis set ("priced in") depending on the residual cost associated with them.

Branch-and-reduce Ryoo and Sahinidis (1996), implemented in the BARON solver (Sahinidis, 1996) also performs automated bounds tightening.

Although these methods lead to more efficient solution than exhaustive brute-force search, they do not take full advantage of the problem structure and require solving numerous subproblems each of similar complexity to the best dynamic programming approaches. They remain useful approaches for MILP and MIQP such as the hinging hyperplane fitting algorithm of Roll et al. (2004) and Toriello and Vielma (2012)'s models for grid tesselation using triangles and surface fitting using convex piecewise-affine functions.

For even more difficult mixed-integer programs, where objective and/or constraints are non-convex, bounds tightening is still a desirable approach. Belotti et al. (2009) describes use of a convex hull around non-linear constraints to define a relaxed sub-problem. A variation on this theme is offered by Leyffer et al. (2008): use of piecewise polyhedral envelopes; these enveloping polyhedrons are rebuilt using increased detail as the search region becomes reduced via pruning. Unlike Belotti et al.'s convex hull approach, the polyhedrons are not necessarily convex. Neither of these techniques are needed for variation cardinality minimization using integral-square residual error. However, if the goodness-of-fit formula were changed to a non-convex function, perhaps to account for a bimodal noise distribution, then such methods would be needed.

1.3.6. Related Work in Dynamic Programming

1.3.6.1. Curve Estimation in the Dynamic Programming Literature. Dynamic programming found applications to piecewise curve estimation quite early. Bellman (1961) described a simple approach to finding the sequence of N disjoint segments that most closely approximated, in a minimum integral-square residual error sense, an integrable function over a fixed interval. For this method the abscissa of each end-point is constrained to a discretization of the interval. The state space was defined by the coordinate pair (right edge of interval, number of segments used). The computational complexity therefore is $\mathcal{O}(u_N^2 N)$ comparisons, where u_N is the number of discrete steps quantizing the interval. If memoization is used, u_N coefficient-search steps may be performed in $\mathcal{O}(u_N)$ time.

Continuity between segments was considered in Bellman and Roth (1969), where a polyline of N (joined) segments is used to approximate an arbitrary continuous curve, where all join points are required to lie on lattice points (both abscissa and ordinate are quantized). The state space is the triple (abscissa of end point, ordinate of end point, number of segments used) leading to a complexity of $\mathcal{O}(h^2 v^2 N)$ comparisons, where ordinate and abscissa are quantized into v and h discrete steps, respectively. Bellman and Roth uses a somewhat unusual cost function, the sum of per-segment maximum absolute error.

Both these algorithms exhibit poor scaling, but motivated further use of dynamic programming.

Another pioneer in use of curve fitting via dynamic programming was Guthery (1974), whose notation of k for the number of partitions and n for the number of data samples, shall be used going forward due to its clarity. He first offers the insight that, given sufficient space, memoization reduces the number of coefficient-search steps in the disjoint piecewise estimation problem to $\mathcal{O}(n^2)$ from $\mathcal{O}(n^2 k)$ (and the number of residual calculations in the piecewise-continuous problem is likewise reduced by a factor of k), although the number of comparisons remains linear in N. Then Guthery introduces Partition Regression, which estimates models for sampled data using piecewise parameterized curves. The first-order auto-regressive models used for curve segments permit dynamics which are exponential in time and non-linear in the coefficients. The method is unsuitable for high sample rate data, not because of scaling complexity, which is quadratic in number of samples, but because the pairwise treatment of data amplifies high frequency noise. Use of higher order auto-regression models is clearly possible but the modeling of individual segments might then become expensive. Recognizing that combining the individual partition models into a single continuous curve might be sub-optimal, Guthery recommends adopting the selected partition boundaries and performing concurrent reoptimization of the parameters.

1.3.6.2. Reception of Digital Codes. Maximum likelihood recovery of signals from noisy measurements using log-likelihood as the goodness-of-fit metric and exploration of a tree dates to Fano's sequential decoder. In the sequential decoder (Fano, 1963), which is applicable to digital signals which are members of a discrete set of levels and with a known symbol duration, the n-ary tree representing the signals is explored depth-first, with one one branch fully expanded, and backtracking is used to trigger exploration of other branches based on a mutual information heuristic. The backtracking distance is bounded either by finite memory in the decoder, or the desire to make a decision after a finite delay.

In his asymptotically optimal decoding algorithm for convolutional codes, Viterbi (1967) guaranteed optimality while eliminating the dependence of time and storage requirements on the input data. The linear feedback shift register defines a Markov chain, because the next state of the register depends only on the prior state and the transmitted signal, without regard for the specifics of the historical path leading to that state (Markov property). The algorithm visits each transition on the Markov chain, computing the likelihood of each path from the likelihood of the "survivor" path associated with the prior state and the correlation of the possible transmission with the actual received signal. Then as a consequence of the Markov property, only the highest-likelihood path terminating in each state is preserved, becoming the new "survivor"

path for that state. (Highest likelihood corresponds to dominance unless the noise properties of the channel are correlated in ways that Viterbi termed "pathological") Because the number of survivor paths is limited by the number of encoder states, the storage requirements do not depend on signal length, and decoding computational requirements are linear in signal length.

Although they did not use the term "memoization", the technique was applied by both Fano and Viterbi, who noted that the log-likelihood of independent symbols shared terms between multiple paths, and computed metrics using the partial sums and new samples. And Viterbi's survivor-selection is, in all but name, Bellman's "Principle of Optimality".

In these decoders the timing is assumed to be known. More likely is the case when the time between steps is known but the time of the first bit is not, in which case clock recovery is employed to determine the phase, and the decoding results are processed by a synchronizer to find which symbol is the first in a message. Clock recovery circuits may themselves employ maximum likelihood techniques (Kobayashi, 1971).

1.3.6.3. Pruned dynamic programming. While the state enumeration methods used by the Viterbi decoder and subsolution enumeration for curve fitting on a lattice (Bellman and Roth, 1969) are powerful for guaranteeing optimality, they require imposing coarse quantization on the signal to restrict the number of potential states.

The approximation method used by Bellman (1961) and later given the name "Segment Neighborhood" eliminates quantization error by changing the ordinate from part of the subsolution identifier to an attribute of the solution. The subsolutions are instead identified by the number of input data points included and the number of segments used. Computation requires $\mathcal{O}(kn^2)$ cost evaluations. For analysis in order of data arrival, the storage requirement is for $\mathcal{O}(kn)$ subsolutions identified by sample index and number of segments. For retrospective analysis when the iteration order can be reversed, only $\mathcal{O}(n)$ subsolutions need to be stored.

Rigaill (2010) improves this situation considerably, by making the abscissa part an attribute as well, and identifying subsolutions by number of segments used and set of ordinate intervals over which the particular subsolution is optimal. When the optimal set for any prior subsolution becomes empty, that subsolution is dropped from consideration. In this "Pruned Dynamic Programming" algorithm, the total number of intervals which must be tracked is observed to remain relatively constant in many realistic datasets, although degenerate cases such as monotonically increasing measurements yield no pruning and once again lead to $\mathcal{O}(kn)$ growth in the number of stored subsolutions. Rigaill completely commits to in-order data processing and notes that this permits real-time applications, although no evaluation of online estimation error performance is conducted.

Meanwhile, Yao (1984) showed that when the likelihood of a change in any single sample interval is time-invariant (leading to geometrically distributed segment durations) and the relationship between its probability and the probability distribution of noise is known, then recursion and induction may be applied (reinventing dynamic programming) to find the maximum likelihood signal estimate in $\mathcal{O}(n^2)$ steps and $\mathcal{O}(n)$ storage, with no dependence on the number of segments. This method was later given the name "Optimal Partitioning".

Killick et al. (2012) improves on Optimal Partitioning by using a variable-sized set of subsolutions like Rigaill. Unlike Rigaill's Pruned DP, this is done without changing addressing of subsolutions – the time of the last change remains the unique identifier of a subsolution. Instead, a dominance condition is implemented leaving zero "survivor" paths at some addresses. Killick et al. provides statistical conditions on segment count and spacing under which the number of survivor paths is bounded and the overall computation time is $\mathcal{O}(n)$, earning the name "Pruned Exact Linear Time" algorithm.

1.4. Real-Time Implications

The greedy chi-squared minimization and total variation minimization algorithms have excellent time complexity. However, both require the entire data set to be available; the final iteration of residual chi-squared minimization can introduce a step anywhere in the entire dataset, while each iteration of total variation minimization updates the entire estimate vector. Therefore, to use these algorithms in near real-time requires running the entire $\mathcal{O}(n)$ algorithm at each of $\mathcal{O}(n)$ data points. Note that convergence speed of total variation minimization is improved by using the prior result as an initial value, but this only lowers the constant factor – running the algorithm online still has quadratic overall complexity.

Integer Programming methods would be extremely expensive to repeat on an ever-expanding dataset. The optimal solution to the prefix data would lead directly to a known feasible solution and bound on the optimal cost; however due to the weakness of the relaxation, a large search space would still need to be explored.

The previously discussed dynamic programming methods are capable of processing data in-order, so running them online during measurement is efficient. However the Rigaill (2010) pruned DP method requires that the number of segments be bounded a-priori. Allowing both incoming data and growth in the number of segments would be prohibitive in storage complexity. Yet not knowing how many segments to reserve for fitting future changepoints precludes producing estimates in real-time with a fixed bound on segment count, since which of the currently-optimal (for different segment counts) solutions to use cannot be determined.

Killick et al. (2012)'s PELT method lends itself very well to in-order evaluation on causal datasets. Indeed, the simplest form of the VIVA algorithm, although independently developed using a different proof, is an instantiation of PELT. However PELT is inapplicable to connected piecewise-linear estimation and also lacks some refinements that will be shown to reduce online estimation error.

Since none of this prior work showed any results for running their step-fitting algorithms on partial datasets, the delay in identifying steps in newly collected data using existing methods remains unknown.

CHAPTER 2

Short History of the Project

When we started using load cells at the UTMB Resuscitation Research Lab, one goal was to overcome the impracticality of deploying large numbers of ultrasonic flowmeters, I tried using traditional low-pass filters to reject the measurement noise, but it quickly became apparent that using frequency domain filtering would not be viable. The level of filtering needed to reduce noise sufficiently for flow rate calculations with acceptable accuracy would both introduce many minutes of group delay and also blur control events such as start and stop of bolus unacceptably. Because the Boyd and Vandenberghe text described preservation of step changes as one of the key advantages of total variation denoising, I determined to try it with the infusion data. I intended to implement it myself to permit customization, but in order both to determine how effective it would be and to serve as a known good result to use for testing my code, I first downloaded and ran an implementation, `tvdip`, written by Little and Jones (2010).

After running `tvdip` on several of my infusion data sets, I observed that, with appropriate values of the λ parameter, it yielded results that were stable and noise-free. However, the rate resulting from processing short duration boluses was consistently underreported. After considering the total variation minimization cost function, I concluded that this was a result of penalizing the size of variations. If instead the cost function penalized residual error and *cardinality* of variations, the optimum solution would become an unbiased estimate. Unfortunately Boyd and Vandenberghe also detailed reasons why using ℓ^0 norm regularization for finding the sparsest solution was intractable and engineers use the ℓ^1 norm for basis pursuit instead. This led to the concept of using `tvdip` to identify timing, and use linear regression for the flow rate in each interval. But I had another idea as well.

Viterbi decoding (Viterbi, 1967) has intrigued me ever since I first encountered its use with convolution code lattices in my undergraduate Communication Systems course. I previously considered using it for ECG beat classification; the limited number of progressions through the cardiac cycle have some similarities to the possible sequences which a linear feedback shift register can generate. Two factors prevented me from developing that idea: First, that it is a very saturated research topic with large amounts of corporate funding, and second, that it didn't align with the primary research goals of the lab.

It was natural, then, that when faced with denoising flow rate signals, I should consider my favorite optimum estimator. While a Viterbi-like approach would be key to efficient exploration of what Boyd and Vandenberghe recognized as essentially an exponential number of possibilities for a binary sequence representing basis membership, one key feature of convolutional codes was missing: Viterbi relies on mapping potential signals onto a finite Markov chain, with multiple sequences ending in exactly the same state, then discarding all but the best. With piecewise curve fitting, the number of possible states is essentially unbounded, as it must summarize the history in a way that satisfies the Markov condition. Clearly, then, domination of multiple paths terminating in a common state would not be possible.

Still, binary programs such as basis pursuit, and mixed-integer programs in general, are not doomed to brute-force search. The technique of branch-and-bound rapidly reduces the search space which must be enumerated. Better still, the branch-and-bound technique allows me to choose the order in which variables are fixed, giving me the flexibility to process data in order of

arrival. The solution to the relaxed problem is also trivial (Theorem 2), and its cost is independent of the actual values of future data. Still, because this solution was not feasible, it had limited utility toward reducing the search space.

Another insight was needed to achieve a tractable method of computation: Just as the regularization parameter represents the cost including a jump between levels in the piecewise constant curve fit, that same parameter is the cost of jumping between states of the Markov chain. The feasible solution that had not been revealed by the usual method of linear constraint relaxation would now be provided by the unknown global optimum curve itself. The current lowest-cost path could be spliced to the latter portion of the optimum curve to create a feasible solution and therefore a bound (Theorem 1).

To understand the implications on the size of the search space, it is helpful to return to the perspective of Viterbi decoding on a Markov chain. While the states reached by extending curve segments scatter throughout a state space that would be infinite save for the precision-limited digital representation of measurements, there is a single state encompassing all solutions which include a discontinuity at the current instant. Of these solutions, only one needs to be kept, the rest are dominated and discarded. Therefore the population of solutions grows by at most one each time step, and a worst-case complexity of $\mathcal{O}(n^2)$. The paths being extended maintaining continuity are also subject to being pruned, but domination requires a cost disadvantage equal to the discontinuity cost. Again, this complexity is achieved for online processing which produces estimates during data processing.

Note that the above discussion was presented in terms of piecewise constant signals, but actually applies broadly to the general case of fitting piecewise disconnected curve segments. (Certain additional conditions must be met which enable splicing – the jump between segments needs to be unlimited, and any pre-truncation of a legal segment needs to result in a legal segment of lower cost. This disallows, for example, a minimum segment duration constraint.) For models requiring connected segments, such as the piecewise linear curve fits described in section 3.4, splicing arbitrary paths requires twice the segmentation cost, to form a zero-width segment bridging the gap.

The foundations of VIVA are therefore:

- Explore an exponentially-large space of binary sequences in the order of data arrival.
- Express fit and residual error of individual segments recursively using a low dimensional state, to avoid reprocessing prior samples of the noisy measured signal.
- Prune candidate solutions which are inferior to a spliced solution using the lowest cost candidate

To this base algorithm are added several refinements.

The best live estimate generated online is faulty. The criterion developed in subsection 5.2.1 provides that true segment boundaries will be passed over as sub-optimal until sufficient evidence exists from both sides of the boundary. Therefore it is useful to report k-step behind estimates in near real-time. The same criterion relates the time lag, step resolving power, and regularization parameter. I compute and compare k-step behind estimates for several lag values.

To address burst noise, I noted that periods where the noise is minimal are more trustworthy than during burst noise. To weight the curve fit in favor of low noise regions without relying on external information, I introduced a confidence measure into the norm of the residual error, weighting each measurement inversely proportional to the local variance of the measured data, a concept borrowed from the noise variance measure used in Kalman filtering.

I also perform additional pruning of candidate solutions in the connected piecewise-linear case. These aggressive heuristics sacrifice the guarantee of finding the true optimum in exchange for greatly reduced runtime. Performance testing provides the justification for this tradeoff.

The following material develops the mathematical underpinnings of the VIVA method, presents the denoising algorithm itself, and tests its performance using synthesized data and datasets recorded during actual porcine studies.

CHAPTER 3

Piecewise Minimum Cardinality Curve Fitting using Pruned Dynamic Programming

The ℓ^0 bicriterion regularization formulation for variation cardinality minimization is restated and illustrated. Graphical and mathematical representations of the splicing argument are presented. It is shown that application of the splicing argument to piecewise-constant curve fitting leads directly to the same pruning condition and dynamic programming algorithm used in the PELT method of Killick et al. (2012), but the splicing argument also provides a corresponding pruning condition to piecewise curve fitting with continuity constraints.

An automatic weighting function based on the local variance of the sampled data is developed. This weighting function improves signal recovery in the presence of burst noise.

Two heuristics for truncated search are introduced which provide near-optimal estimation of connected piecewise-linear curve fits, with the same complexity as the piecewise disconnected case (linear, if the PELT conditions on segment duration are met).

Introduction of a limited time delay, comparable to the online delay of traditional linear and median filtering methods, is added.

The weighting function, truncated search method, and delay are validated empirically using a synthesized dataset for Monte-Carlo analysis as well as infusion data from trauma and burn resuscitation research studies in a porcine model. Results for a financial dataset appeared above (Figure 3, Table 1).

3.1. Optimality Criterion for Continuity-Constrained Piecewise Curve Fitting

Development of an optimality criterion using the splicing argument will be demonstrated first using the simplest case, piecewise-constant curve fitting, before moving on to more complex models.

Recall the (mixed-integer quadratic program) model augmented by binary variables identifying changepoints, Equation 1.7.

$$
\begin{array}{lrcccl}
\text{minimize} & \multicolumn{4}{c}{(\mathbf{x}'-\mathbf{y})^T(\mathbf{x}'-\mathbf{y})+\zeta\mathbf{1}^T\mathbf{b}'} & \\
\text{subject to} & -x'_k + x'_{k+1} - Mb'_k & \leq & 0 & & \forall k \\
& x'_k - x'_{k+1} - Mb'_k & \leq & 0 & & \forall k \\
& \multicolumn{4}{c}{b'_k \in \{0,1\}} & \forall k
\end{array}
\tag{3.1}
$$

The following bound on prefix path deficit is known as a necessary condition for optimality due to Killick et al. (2012), who present this pruning criterion after using independence between segments to apply Bellman's Principle of Optimality. The following approach does not rely on independence between segments, so it is not limited to fitting of disjoint curves.

Theorem 1. *If $\mathbf{x}^*, \mathbf{b}^*$ is any optimal solution to Equation 1.7, then the prefix cost along the path*

$$
\begin{aligned}
c^*_\kappa &= c_{\le\kappa}(\mathbf{x}^\dagger, \mathbf{b}^*) \\
&= \sum_{k=0}^{\kappa} \left(x^\dagger_k - y_k\right)^2 + \zeta \sum_{k=0}^{\kappa-1} b^*_k
\end{aligned} \tag{3.2}
$$

is bounded by the path costs of all feasible solutions $\mathbf{x}', \mathbf{b}'$*:*

$$
c^*_\kappa \le c_{\le\kappa}(\mathbf{x}', \mathbf{b}') + \zeta(1 - b^*_\kappa), \quad \forall \kappa, \mathbf{x}', \mathbf{b}' \tag{3.3}
$$

Proof. Construct the spliced solution

$$
x^s_k = \begin{cases} x'_k & k \le \kappa \\ x^*_k & k > \kappa \end{cases} \tag{3.4a}
$$

$$
b^s_k = \begin{cases} b'_k & k < \kappa \\ 1 & k = \kappa \\ b^*_k & k > \kappa \end{cases} \tag{3.4b}
$$

This solution $\mathbf{x}^s, \mathbf{b}^s$ is also feasible for Equation 1.7, with total cost

$$
\begin{aligned}
c^s &= \sum_{k=0}^{N} (x^s_k - y_k)^2 + \zeta \sum_{k=0}^{N-1} b^s_k \\
&= \sum_{k=0}^{\kappa} (x^s_k - y_k)^2 + \sum_{k=\kappa+1}^{N} (x^s_k - y_k)^2 + \zeta \sum_{k=0}^{\kappa-1} b^s_k + \zeta b^s_\kappa + \zeta \sum_{k=\kappa+1}^{N-1} b^s_k \\
&= \sum_{k=0}^{\kappa} (x'_k - y_k)^2 + \zeta \sum_{k=0}^{\kappa-1} b'_k + \sum_{k=\kappa+1}^{N} (x^*_k - y_k)^2 + \zeta + \zeta \sum_{k=\kappa+1}^{N-1} b^*_k \\
&= c_{\le\kappa}(\mathbf{x}', \mathbf{b}') + \zeta + \sum_{k=\kappa+1}^{N} (x^*_k - y_k)^2 + \zeta \sum_{k=\kappa+1}^{N-1} b^*_k
\end{aligned} \tag{3.5}
$$

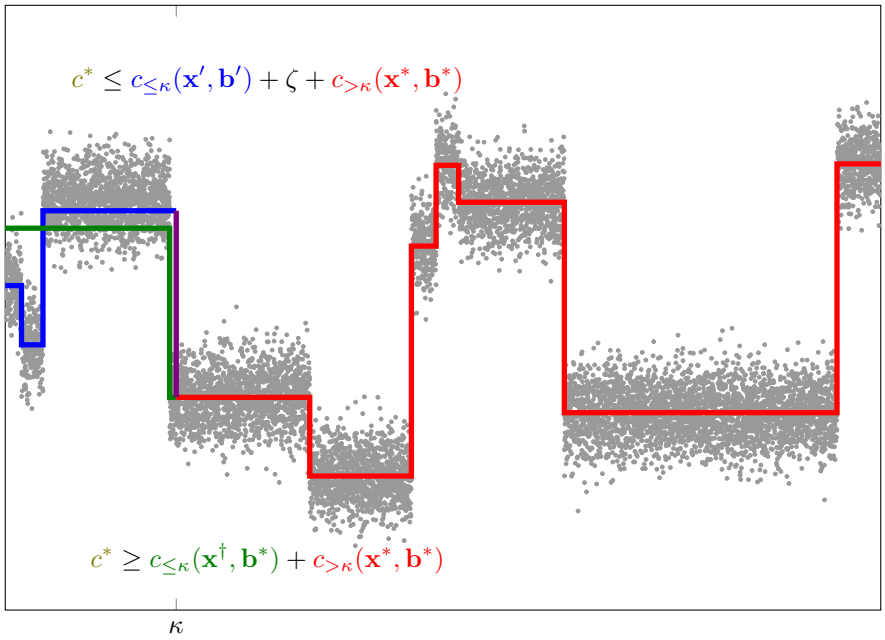
The optimal cost is

$$
\begin{aligned}
c^* &= \sum_{k=0}^{N} (x^*_k - y_k)^2 + \zeta \sum_{k=0}^{N-1} b^*_k \\
&= \sum_{k=0}^{\kappa} (x^*_k - y_k)^2 + \sum_{k=\kappa+1}^{N} (x^*_k - y_k)^2 + \zeta \sum_{k=0}^{\kappa-1} b^*_k + \zeta b^*_\kappa + \zeta \sum_{k=\kappa+1}^{N-1} b^*_k \\
&\ge \sum_{k=0}^{\kappa} \left(x^\dagger_k - y_k\right)^2 + \zeta \sum_{k=0}^{\kappa-1} b^*_k + \sum_{k=\kappa+1}^{N} (x^*_k - y_k)^2 + \zeta b^*_\kappa + \zeta \sum_{k=\kappa+1}^{N-1} b^*_k \\
&= c^*_\kappa + \zeta b^*_\kappa + \sum_{k=\kappa+1}^{N} (x^*_k - y_k)^2 + \zeta \sum_{k=\kappa+1}^{N-1} b'_k
\end{aligned} \tag{3.6}
$$

noting that $x^\dagger_k$ differs from x^*_k only in the segment wherein κ lies, for which x^*_κ is the minimizer over the whole segment, while $x^\dagger_\kappa$ is the minimizer over the truncated segment.

By the optimality of $\mathbf{x}^*, \mathbf{b}^*$,

$$
c^* \le c^s \tag{3.7a}
$$



$$\hat{c}_{\le\kappa}(\mathbf{x}^\dagger, \mathbf{b}^*) + c_{>\kappa}(\mathbf{x}^*, \mathbf{b}^*) \le c_{\le\kappa}(\mathbf{x}', \mathbf{b}') + \zeta + c_{>\kappa}(\mathbf{x}^*, \mathbf{b}^*)$$

$$c_{\le\kappa}(\mathbf{x}^\dagger, \mathbf{b}^*) \le c_{\le\kappa}(\mathbf{x}', \mathbf{b}') + \zeta$$

FIGURE 5. Cancel common term, yielding necessary condition for prefix of optimal segmentation

and therefore

(3.7b)

$$c^*_\kappa + \zeta b^*_\kappa + \sum_{k\ =\kappa+1}^{N} \left(x'_k - y_k\right)^2 + \zeta \sum_{k\ =\kappa+1}^{N-1} b'_k \le c_{\le\kappa}(\mathbf{x}', \mathbf{b}') + \zeta + \sum_{k=\kappa+1}^{N} \left(x'_k - y_k\right)^2 + \zeta \sum_{k=\kappa+1}^{N-1} b'_k$$

After cancellation of like terms,

$$c^*_\kappa \le c_{\le\kappa}(\mathbf{x}', \mathbf{b}') + \zeta(1 - b^*_\kappa) \tag{3.8}$$

□

Figure 5 provides a graphical illustration of the path splicing approach. It should be immediately apparent that path splicing is also viable for connected piecewise curves, such as the polyline (piecewise-linear) model Equation 3.9 (given earlier as Equation 1.8).

(3.9)

$$\begin{array}{lllllll}
\text{minimize} & & (\mathbf{x}' - \mathbf{y})^T (\mathbf{x}' - \mathbf{y}) + \zeta \mathbf{1}^T \mathbf{b}' & & & & \\
\text{subject to} & (t_i - t_{i+1})x'_{k-1} & + (t_{i+1} - t_{i-1})x'_k & + (t_{i-1} - t_i)x'_{k+1} & - Mb'_k & \le\ 0 & \forall k \\
& (t_{i+1} - t_i)x'_{k-1} & + (t_{i-1} - t_{i+1})x'_k & + (t_i - t_{i-1})x'_{k+1} & - Mb'_k & \le\ 0 & \forall k \\
& & b'_k \in \{0, 1\} & & & & \forall k
\end{array}$$

Splicing leads to the feasible piecewise-linear solution

$$x_k^s = \begin{cases} x'_k & k \leq \kappa - 1 \\ x_k^* & k \geq \kappa \end{cases} \tag{3.10a}$$

$$b_k^s = \begin{cases} b'_k & k < \kappa - 1 \\ 1 & k = \kappa - 1 \\ 1 & k = \kappa \\ b_k^* & k > \kappa \end{cases} \tag{3.10b}$$

and using the same argument, the optimality condition used to perform pruning during continuous piecewise-linear curve fitting is:

$$c_\kappa^* \leq c_{\leq\kappa}(\mathbf{x}', \mathbf{b}') + \zeta(2 - b_{\kappa-1}^* - b_\kappa^*), \quad \forall \kappa, \mathbf{x}', \mathbf{b}' \tag{3.11}$$

Although the branch-and-bound method cannot be applied directly without a feasible solution to the entire problem, this bound on partial path costs permits pruning suboptimal solutions at each timestep. In particular, in the case of disjoint curve segments, only one candidate with $b_\kappa = 1$ needs to be preserved to achieve optimality. Applying pruning to breadth-first search yields a search algorithm very similar to Viterbi decoding.

3.2. Complexity Growth in Small Cases

For disjoint piecewise curve fits, the pruning criterion guarantees that the population growth cannot exceed one new candidate per time step. Growth in the polyline (or generalized connected curve segments) case, where independence between parameters across changepoints does not apply, remains problematic.

Using the first 30 % of the S&P 500 dataset used in Kim et al. (2009) for trend filtering, dynamic programming was used to find the estimates with k segments, $k \in 2, 3, 4$ having minimum mean squared residual error. The complexity growth of this approach is shown in Figure 6 and the results in Figure 7 and its accompanying table. Identification of the optimal estimate with k segments in a dataset of N points requires analysis of the N points at each of the $\binom{N-2}{k-1}$ possible breakpoints. The total computational cost is somewhat better than $\binom{N-1}{k}$ due to memoization of intermediate results. This agrees with Boyd and Vandenberghe (2004, p. 310).

Bicriterion regularization was also performed using pruned dynamic programming with the optimality criterion (Equation 3.11) and $\zeta = 0.13$, which found the three segment solution. Its complexity growth as a function of N is also shown in Figure 6.

Initially, there appears to be no benefit to using the bicriterion formulation and pruned DP, since the cost was higher (and scaling worse) than the cost of the combinatorial search that found the same result. However, the combinatorial search fails to ensure that the result is an optimum in the bicriterion sense. For example, the four segment path, although a minimal mean-square residual error result, is not optimum for any value of ζ: it dominates the three segment path for $\zeta < 0.0701$, but a five segment solution (1, 7.1732)–(356, 7.2989)–(479, 7.1659)–(511, 7.0211)–(531, 7.1565)–(600, 7.0796) dominates both for $\zeta < 0.0755$. Using combinatorial methods to show that three segments are optimal for $\zeta = 0.13$ would require evaluating $\left\lfloor \frac{MMSE}{\zeta} \right\rfloor$ further cases – up to a seven segment polyline – at the cost of substantial additional computation.

Additional comparisons of exact polyline fitting using combinatorial methods and bicriterion regularization via pruned dynamic programming may be found in Appendix E.

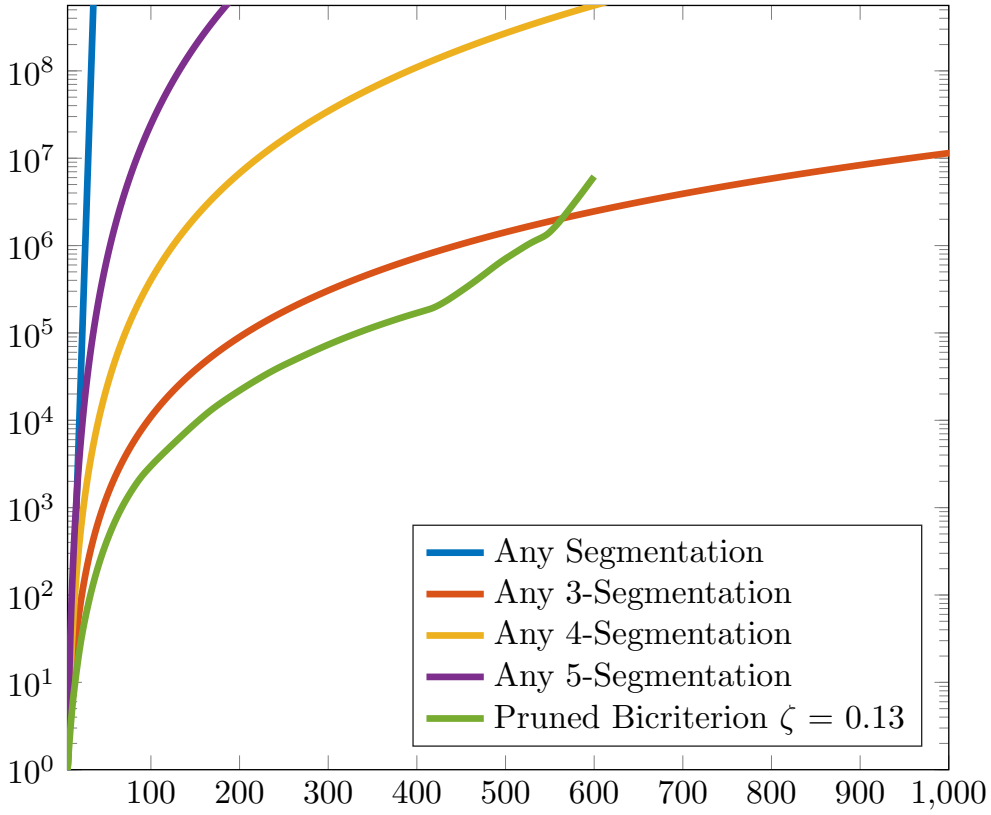


FIGURE 6. Exact polyline search algorithms' complexity, considering first 600 data of S&P 500 log(price) data (Kim et al., 2009)

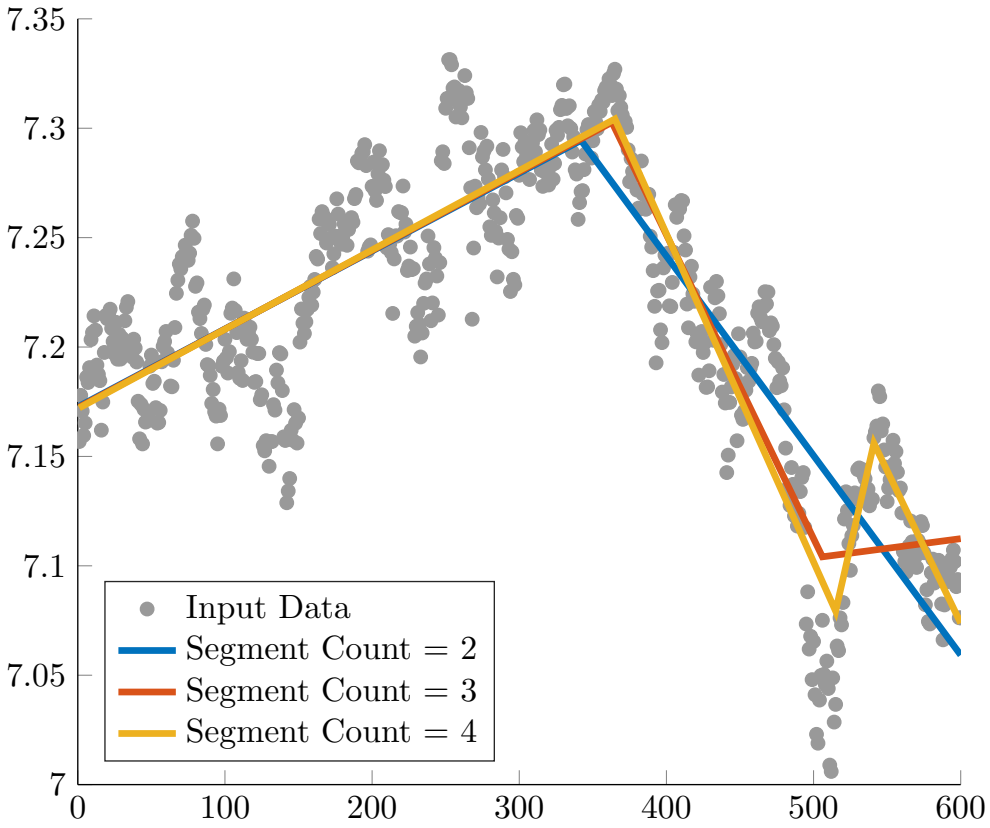


FIGURE 7. Optimal polyline approximations for first 600 samples of S&P 500 log(price) data (Kim et al., 2009)

TABLE 2. Residual error and optimality, by segment count, for the S&P 500 log(price) excerpt polyline approximations

Segments	Solution	Cost	Optimal When
2	(1, 7.1730) – (342, 7.2944) – (600, 7.0592)	$0.7257 + \zeta$	$0.15 < \zeta < 1.71$
3	(1, 7.1726) – (363, 7.3027) – (506, 7.1041) – (600, 7.1124)	$0.5794 + 2\zeta$	$0.07 < \zeta < 0.15$
4	(1, 7.1722) – (365, 7.3043) – (515, 7.0793) – (541, 7.1560) – (600, 7.0738)	$0.5093 + 3\zeta$	Never

3.3. Functional Description of VIVA Algorithm

3.3.1. Dense State Encoding. Implementation of the VIVA algorithm leverages the least-squares fit and residual expressions developed in Appendix B and recursive update techniques given in Appendix C.2.

Define a function for highest element set in a binary sequence:

$$h(\mathbf{b}, n) = \begin{cases} -1 & b_n = 0 \cap n = 0 \\ h(\mathbf{b}, n-1) & b_n = 0 \cap n > 0 \\ n & b_n = 1 \end{cases} \tag{3.12}$$

Using the usual definition of list:

$$\text{list}\,'T = \text{Nil} \mid 'T :: \text{list}\,'T \tag{3.13}$$

Define a state-space for candidate solutions, consisting of a 5-tuple:

$$\mathrm{A} = \mathbb{R} \star \mathbb{R}_+ \star \mathbb{R}_+ \star \mathbb{Z}_{++} \star (\text{list}\,(\mathbb{Z}_{++} \star \mathbb{R})) \tag{3.14}$$

Each state is a dense representation at time κ of a candidate solution $\mathbf{x}, \mathbf{b}$ evaluated using input data $\mathbf{y}$ with the invariant

$$\begin{aligned} \kappa \star \mathbf{x}' \star \mathbf{b}' \to & \left(\sum_{k=h(\mathbf{b}')+1}^{\kappa-1} y_k \right) \star \left(\sum_{k=h(\mathbf{b}')+1}^{\kappa-1} y_k^2 \right) \star \left(\sum_{k=0}^{h(\mathbf{b}')} (x'_k - y_k)^2 + \zeta \sum_{k=0}^{h(\mathbf{b}')} b'_k \right) \\ & \star (\kappa - h(\mathbf{b}')) \star \left((\Delta\tilde{t}_i \star \tilde{x}_i) :: \cdots :: (\Delta\tilde{t}_0 \star \tilde{x}_0) \right) \end{aligned} \tag{3.15}$$

Accessor functions give symbolic names to the components of the tuple:

$$\begin{aligned} \text{sum} \;:\; & \mathrm{A} \to \mathbb{R} \\ & \textbf{is} \quad (\text{first} : \mathbb{R}) \star (\mathbb{R}_+) \star (\mathbb{R}_+) \star (\mathbb{Z}_{++}) \star (\text{list}\,(\mathbb{Z}_{++} \star \mathbb{R}) \to \text{first} \end{aligned} \tag{3.16a}$$

$$\begin{aligned} \text{energy} \;:\; & \mathrm{A} \to \mathbb{R}_+ \\ & \textbf{is} \quad (\mathbb{R}) \star (\text{second} : \mathbb{R}_+) \star (\mathbb{R}_+) \star (\mathbb{Z}_{++}) \star (\text{list}\,(\mathbb{Z}_{++} \star \mathbb{R}) \to \text{second} \end{aligned} \tag{3.16b}$$

$$\begin{aligned} \text{path\_cost} \;:\; & \mathrm{A} \to \mathbb{R}_+ \\ & \textbf{is} \quad (\mathbb{R}) \star (\mathbb{R}_+) \star (\text{third} : \mathbb{R}_+) \star (\mathbb{Z}_{++}) \star (\text{list}\,(\mathbb{Z}_{++} \star \mathbb{R}) \to \text{third} \end{aligned} \tag{3.16c}$$

$$\begin{aligned} \text{age} \;:\; & \mathrm{A} \to \mathbb{Z}_{++} \\ & \textbf{is} \quad (\mathbb{R}) \star (\mathbb{R}_+) \star (\mathbb{R}_+) \star (\text{fourth} : \mathbb{Z}_{++}) \star (\text{list}\,(\mathbb{Z}_{++} \star \mathbb{R}) \to \text{fourth} \end{aligned} \tag{3.16d}$$

$$\begin{aligned} \text{path} \;:\; & \mathrm{A} \to \text{list}\,(\mathbb{Z}_{++} \star \mathbb{R}) \\ & \textbf{is} \quad (\mathbb{R}) \star (\mathbb{R}_+) \star (\mathbb{R}_+) \star (\mathbb{Z}_{++}) \star (\text{fifth} : \text{list}\,(\mathbb{Z}_{++} \star \mathbb{R}) \to \text{fifth} \end{aligned} \tag{3.16e}$$

In particular, this state representation allows simple computation of the cost $c_{\leq\kappa}(\mathbf{x}, \mathbf{b})$

$$\begin{aligned} \text{cost} \;:\; & \mathrm{A} \to \mathbb{R}_+ \\ & \textbf{is} \quad (a : \mathrm{A}) \to (\text{path\_cost}\ a) + (\text{energy}\ a) - \tfrac{(\text{sum}\ a)^2}{\text{age}\ a} \end{aligned} \tag{3.17}$$

Functions for inclusion of one additional sample (in arrival order) are straightforward.

$$\begin{aligned} \text{branch0} \;:\; & \mathbb{R} \to \mathrm{A} \to \mathrm{A} \\ & \textbf{is} \quad (\text{sample} : \mathbb{R}) \to (a : \mathrm{A}) \to ((\text{sum}\ a) + \text{sample}) \star ((\text{energy}\ a) + \text{sample}^2) \\ & \qquad \star (\text{path\_cost}\ a) \star ((\text{age}\ a) + 1) \star (\text{path}\ a) \end{aligned} \tag{3.18}$$

$$\begin{aligned} \text{branch1} \;:\; & \mathbb{R} \to \mathrm{A} \to \mathrm{A} \\ & \textbf{is} \quad (\text{sample} : \mathbb{R}) \to (a : \mathrm{A}) \to (\text{sample}) \star (\text{sample}^2) \star (\text{cost}\ a + \zeta) \star (1) \\ & \qquad \star (\textbf{cons}\ ((\text{age}\ a) \star (\tfrac{\text{sum}\ a}{\text{age}\ a}))\ (\text{path}\ a)) \end{aligned} \tag{3.19}$$

Several other useful list operations:

(3.20a)

$$\begin{aligned}&\text{reduce} : {}'S \star ({}'S \star {}'T \to {}'S) \star \text{list } {}'T \to {}'S\\ &\quad \textbf{is} \quad (\text{accum} : {}'S) \star ({}'S \star {}'T \to {}'S) \star \text{Nil} \to \text{accum}\\ &\qquad\quad (\text{accum} : {}'S) \star (\text{op} : {}'S \star {}'T \to {}'S) \star ((\text{head} : {}'T) :: (\text{tail} : \text{list } {}'T))\\ &\qquad\qquad\qquad\qquad \to \text{op head (reduce accum op tail)}\end{aligned}$$

(3.20b)

$$\begin{aligned}&\text{min} : ({}'T \to \mathbb{R}) \star \text{list } {}'T \to {}'T\\ &\quad \textbf{is} \quad (\text{selector} : {}'T \to \mathbb{R}) \star ((\text{head} : {}'T) :: (\text{tail} : \text{list } {}'T))\\ &\qquad \to \text{reduce head } (\lambda\text{best}.\lambda\text{elem}.(\textbf{if } ((\text{selector elem}) < (\text{selector best})) \text{ elem best})) \text{ tail}\end{aligned}$$

(3.20c)

$$\begin{aligned}&\text{map\_filter}\\ &\quad : ({}'T \to \text{Bool}) \star ({}'T \to {}'T) \star \text{list } {}'T \to \text{list } {}'T\\ &\quad\ \textbf{is} \quad ({}'T \to \text{Bool}) \star ({}'T \to {}'T) \star \text{Nil} \to \text{Nil}\\ &\qquad\quad (\text{pred} : {}'T \to \text{Bool}) \star (f : {}'T \to {}'T) \star (\text{items} : \text{list } {}'T)\\ &\qquad\qquad\qquad\qquad \to \textbf{let } \text{selected} : \text{Bool} = \text{pred (head items) } \textbf{in}\\ &\qquad\qquad\qquad\qquad\quad \textbf{let } \text{rest} : \text{list } {}'T = (\text{map\_filter pred } f \text{ (tail items)}) \textbf{ in}\\ &\qquad\qquad\qquad\qquad\quad \textbf{if } \text{selected } (\textbf{cons } (f \text{ (head items)}) \text{ rest}) \text{ rest}\end{aligned}$$

The population of candidate solutions will be initialized as

$$P_0 : \text{list A} \Leftarrow \textbf{cons } (\text{branch0 sample } (0 \star 0 \star 0 \star 0 \star \text{Nil})) \text{ Nil} \tag{3.21}$$

At each incoming sample, the population update algorithm is

(3.22)

$$\begin{aligned}&\text{timestep} : \mathbb{R} \star \text{list A} \to \text{list A}\\ &\quad \textbf{is} \quad (\text{sample} : \mathbb{R}) \star (\text{population} : \text{list A})\\ &\qquad \to \textbf{let } \text{min\_cost} : \text{A} = \text{min cost population } \textbf{in}\\ &\qquad\quad \textbf{let } \text{threshold} : \mathbb{R}_+ = \zeta + (\text{cost min\_cost}) \textbf{ in}\\ &\qquad\quad \textbf{cons } (\text{branch1 sample min\_cost})\\ &\qquad\qquad\quad (\text{map\_filter } \lambda a.((\text{cost } a) < \text{threshold}) \text{ (branch0 sample) population})\end{aligned}$$

3.4. Truncated Search Heuristics

Although performing bicriterion regularization using pruned dynamic programming scales better than using combinatorial enumeration of changepoint basis sets, finding exact solutions in the presence of continuity constraints remains prohibitively expensive for larger datasets.

In many cases performing optimal recovery is not warranted, if the difference between optimal and near-optimal fits is insignificant compared to the noise floor resulting from other confounding factors. Heuristics for truncated search seek to trade guaranteed optimality for reduced estimation cost. Two heuristics are proposed which accept a small increase in bicriterion residual but achieve tractability for problems with millions of data points.

The first of these heuristics observes that, in the subproblem of finding optimal polyline vertex ordinates for a fixed sequence of abscissas, measurements have little effect on vertices more than two segments away. By assuming the accumulated effect is trivial, segment vertices, and therefore also the associated residuals, become fixed as additional segments are added to the path. Because the residuals become fixed, so does their ordering, allowing many paths up to a particular segment to be reduced to one as that segment becomes frozen.

The second heuristic admits only one branch per measurement interval.

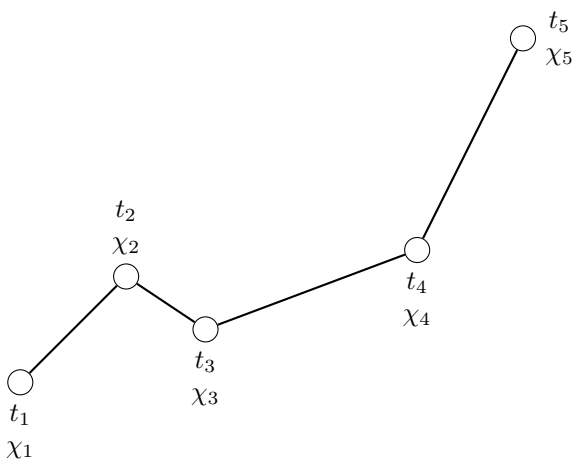


FIGURE 8. Model of many conjoined piecewise-linear segments

3.4.1. "Freeze Old Segments" Heuristic

Consider that the N-segment least-squares optimization step (shown in Figure 8) described by Equation 3.23 can be performed efficiently using the tridiagonal matrix algorithm (Thomas, 1949) which exhibits linear scaling, as does calculation of the residual $\epsilon(\tilde{\chi})$.

$$\begin{bmatrix} d_1 & a_2 & 0 & \cdots & 0 \\ a_2 & d_2 & a_3 & 0 & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & a_{N-1} & d_{N-1} & a_N \\ 0 & \cdots & 0 & a_N & d_N \end{bmatrix} \tilde{\chi} = \begin{bmatrix} S(t_1,t_2) - \dfrac{R(t_1,t_2)}{N(t_1,t_2)} \\ S(t_2,t_3) - \dfrac{R(t_1,t_2)}{N(t_2,t_3)} \\ \vdots \\ S(t_{N-1},t_N) - \dfrac{R(t_{N-1},t_N)}{N(t_{N-1},t_N)} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \dfrac{R(t_1,t_2)}{N(t_1,t_2)} \\ \dfrac{R(t_2,t_3)}{N(t_2,t_3)} \\ \vdots \\ \dfrac{R(t_{N-1},t_N)}{N(t_{N-1},t_N)} \end{bmatrix} \tag{3.23a}$$

where

$$d_k = \begin{cases} 1 & k = 1 \\ \dfrac{(N(t_{k-1},t_k)+1)\,(2N(t_{k-1},t_k)+1)}{6N(t_{k-1},t_k)} & k > 1 \end{cases} + \begin{cases} \dfrac{(N(t_k,t_{k+1})-1)\,(2N(t_k,t_{k+1})-1)}{6N(t_k,t_{k+1})} & k < N \\ 0 & k = N \end{cases} \tag{3.23b}$$

$$a_k = \frac{(N(t_{k-1},t_k)-1)\,(N(t_{k-1},t_k)+1)}{6N(t_{k-1},t_k)} \quad \forall k > 1 \tag{3.23c}$$

and

$$\epsilon(\tilde{\chi}) = \sum_{k=2}^{N} S^2(t_{k-1},t_k) - \tilde{\chi}^T \left(\begin{bmatrix} S(t_1,t_2) - \dfrac{R(t_1,t_2)}{N(t_1,t_2)} \\ S(t_2,t_3) - \dfrac{R(t_1,t_2)}{N(t_2,t_3)} \\ \vdots \\ S(t_{N-1},t_N) - \dfrac{R(t_{N-1},t_N)}{N(t_{N-1},t_N)} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \dfrac{R(t_1,t_2)}{N(t_1,t_2)} \\ \dfrac{R(t_2,t_3)}{N(t_2,t_3)} \\ \vdots \\ \dfrac{R(t_{N-1},t_N)}{N(t_{N-1},t_N)} \end{bmatrix} \right) \tag{3.23d}$$

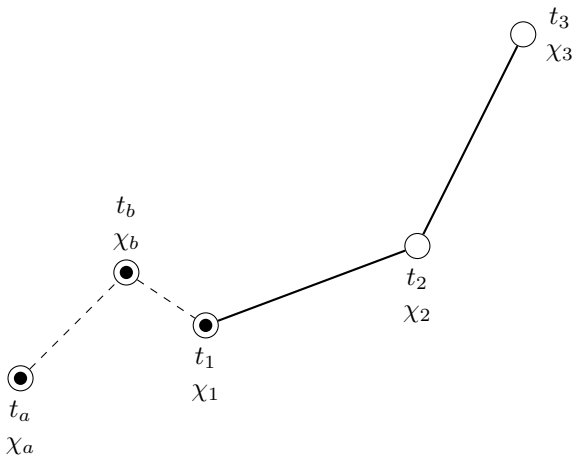


FIGURE 9. Model of many conjoined piecewise-linear segments with segment freeze heuristic

The previous method incurs increased complexity as the number of segments increases. In addition, although arriving data have ever-diminishing impact on the first ramp estimate, the global optimal fit cannot be computed until all data are available.

The VIVA algorithm implements a "freeze old segments" heuristic to sacrifice global optimality for reduced latency. Two ramps are considered, then the first of these is fixed in place and fitting of the second and third ramps proceeds. The method repeats by fixing the first of these (the second ramp overall) and optimizing the two following, and so on. In this manner, all optimization steps concerning the third and subsequent segments have identical form, as illustrated in Figure 9.

A consequence of freezing an old vertex each time a path is branched, to maintain exactly two segments subject to optimization at all times, is that the cumulative cost of frozen segments, and therefore also their ordering, becomes fixed as well. Therefore, although there be many candidate paths permitted to branch by the criterion (3.11), all being frozen at the same time coordinate may be compared and only the best of these allowed to branch. In this way, population growth (branches per timestep) is reduced from combinatorial (all ways of distributing ζ surplus residual error around prior changepoints) to linear.

3.4.2. "Limited Branching" Heuristic

Even with the frozen-path-domination branch generation logic ensuring at most linear population growth, accumulated population size (and therefore memory usage) has a quadratic bound, which may exceed resources available in an embedded environment. Therefore VIVA implements a stronger version of this heuristic as a runtime option, permitting only a constant number of new branches at each timestep (and therefore linear memory usage and quadratic runtime), resulting in modest additional estimation error.

Notably, when the change timing conditions described by Killick et al. (2012) are met, memory usage reduces to $\mathcal{O}(1)$ and runtime to $\mathcal{O}(n)$. This heuristic is particularly well suited to online implementation (see chapter 5) where delay and unavailability of future data, not this heuristic, are the dominant factors contributing to estimation error.

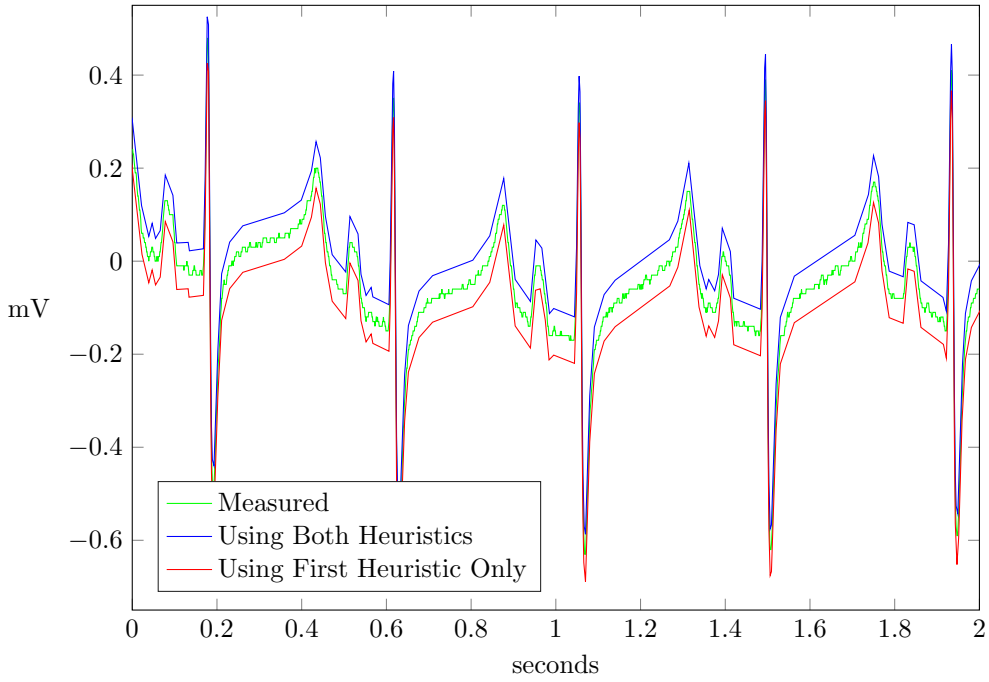


FIGURE 10. Excerpt of ECG data compression example data, with vertical offset added for ease of comparison

3.5. Case Study, Compression of Electrocardiogram Data

Telemedicine allows medical experts to assist in patient care regardless of geographic separation. For the doctor to clearly diagnose and manage the patient, vital signs such as electrocardiograms must be collected and transmitted to the expert. Telemedicine is particularly valuable when local resources are lacking, either because the injury occurred in an austere environment with minimal medical support, or because the local support is overwhelmed by a large number of patients (battlefields and natural disasters both result n mass casualty scenarios). In either case, communication bandwidth may be scarce as well, in which case efficient compression is needed in order to maximize the amount of clinically useful data that can be sent through the link. In addition, hospital staff are increasingly requesting the ability to forward patient data to wireless mobile devices, often on metered data plans.

An excerpt of slightly longer than an hour was taken from an electrocardiogram recording from a swine hemorrhage and resuscitation experiment conducted in the Resuscitation Research Laboratory, Department of Anesthesiology, University of Texas Medical Branch (Galveston, TX). Polyline fitting was conducted using the "Freeze Old Segments" heuristic, alone and in combination with the "Limited Branching" heuristic, to investigate the deviation from optimality introduced by these heuristics. The original recorded and compressed signals are displayed in Figures 10 and 11, and results given in Table 3.

Although the signal was sufficiently large to make exact search infeasible even using pruned dynamic programming, the truncated search methods produced an excellent approximation. In Figure 10, an intentional offset has been introduced into the fitting results to prevent the signals from overlying each other and obscuring the shape of the estimate; exactly how well the signals coincide may be seen in Figure 11.

After developing the polyline estimates using truncated search, the changepoint timings from this estimate were used to perform least-squares minimization, the method suggested for basis

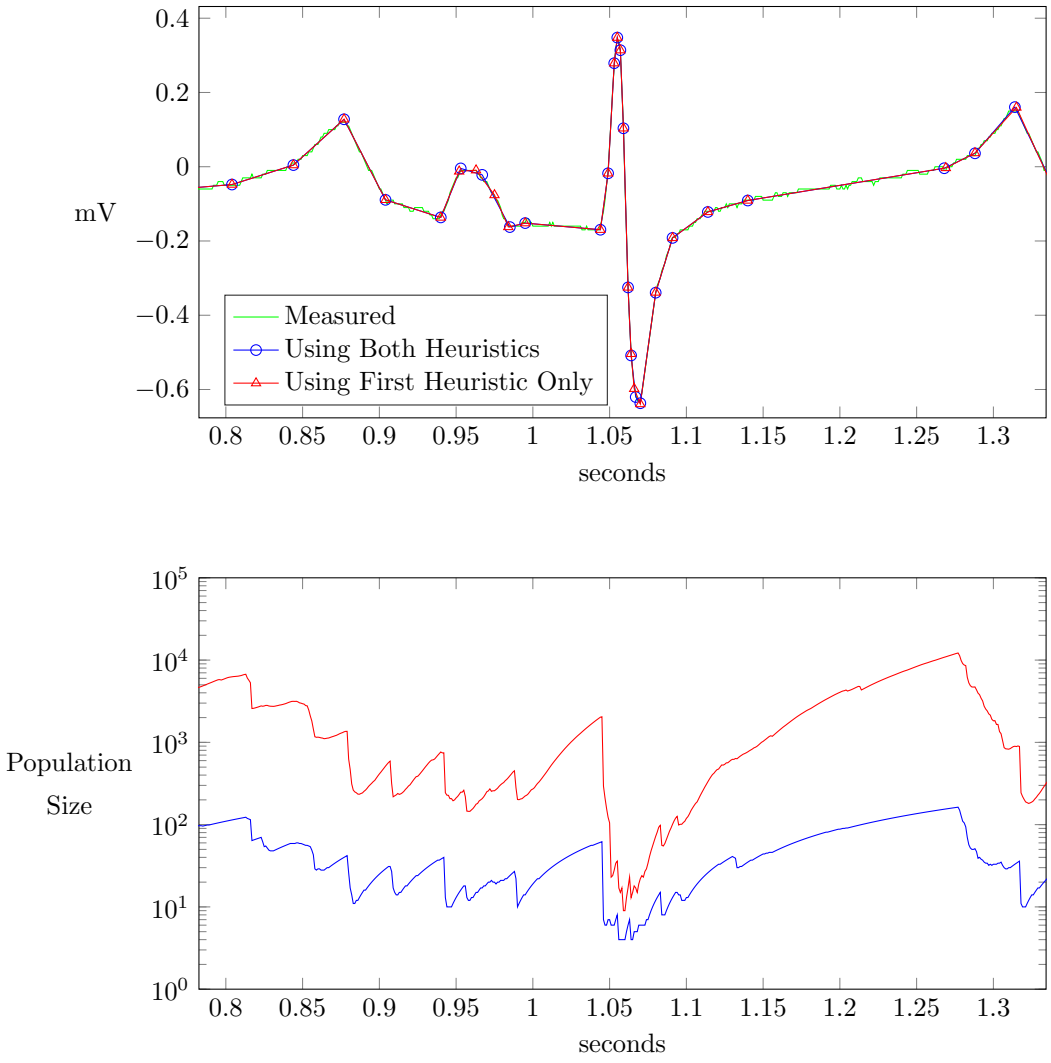


FIGURE 11. Close zoom into ECG data compression example data

pursuit using heuristics in Boyd and Vandenberghe (2004, p. 334). The difference in maximum residual between the global least-squares and the direct results from the VIVA implementation is as much as 5 %, but the RMS residual increases less than 0.5 % with addition of the second heuristic, and varies less than 0.1 % between the direct result and the global least-squares using the same timing. The bicriterion cost metrics are not directly comparable because the VIVA algorithm used variance-based weighting (to be introduced in subsection 4.1.2) and the least-squares minimization did not.

Additional evaluation of complexity advantages of truncated search heuristics vs exact solutions may be found in Appendix E.

TABLE 3. Heuristics' Influence on ECG Polyline Approximation (Compression) Performance

Metric		"Freeze Old Segments" Heuristic	Both Heuristics
Input Sample Rate		1000/s	
Input Sample Count		4 000 001	
QRS Complexes		5809	
Bicriterion Weight ζ		1×10^{-4}	1×10^{-4}
Output Vertex Count		195 654	194 510
Compression Ratio		20.44	20.56
Mean Population Size		3296.7	61.8
Reported Bicriterion Cost[1]		210.0810	210.4589
Global Bicriterion Cost[2]		132.5600	133.3471
Heuristic Residual Max[1]	µV	26.5	25.5
Global Residual Max[2]	µV	25.2	24.9
Heuristic Residual RMS[1]	µV	5.3172	5.3387
Global Residual RMS[2]	µV	5.3150	5.3361

[1] Timing and values obtained from VIVA

[2] Timing obtained from VIVA; values obtained from global least-squares

3.6. Evaluation of Signal Recovery Performance using Synthesized Data

3.6.1. Monte-Carlo Testing Methodology

Studying how the algorithms behave under noxious conditions is useful as a design exercise, but for real-world use it is necessary to evaluate their average performance. For this purpose, and to prevent inadvertently designing inputs to favor a particular algorithm, Monte Carlo simulation has been conducted on a large dataset of randomly generated signals. Scripts used for construction of the signals are provided in Appendix A.

For each record in the input dataset, the competing algorithms process the "measured" signal. The "original" signal parameters and noise are separately known to the test harness, but not provided to the algorithms. The error vector is then computed as the difference between the estimated signal recovered using each algorithm and the original signal. The algorithms are assessed according to the distribution of the originals, characterized by the maximum, 90th percentile, root-mean-square, and mean absolute value metrics. The values are plotted using a logarithmic scale in order to clearly show relative differences at all performance levels.

For piecewise linear signals, both value and rate of change are evaluated. All piecewise-linear results indicated as "VIVA" were obtained using both search truncation heuristics: ordinates were frozen for all except the last two vertices, and only a single branch was allowed at each timestep.

In some cases, a second analysis of the error vector is done, excluding intervals around changepoints, in order to evaluate steady-state error performance.

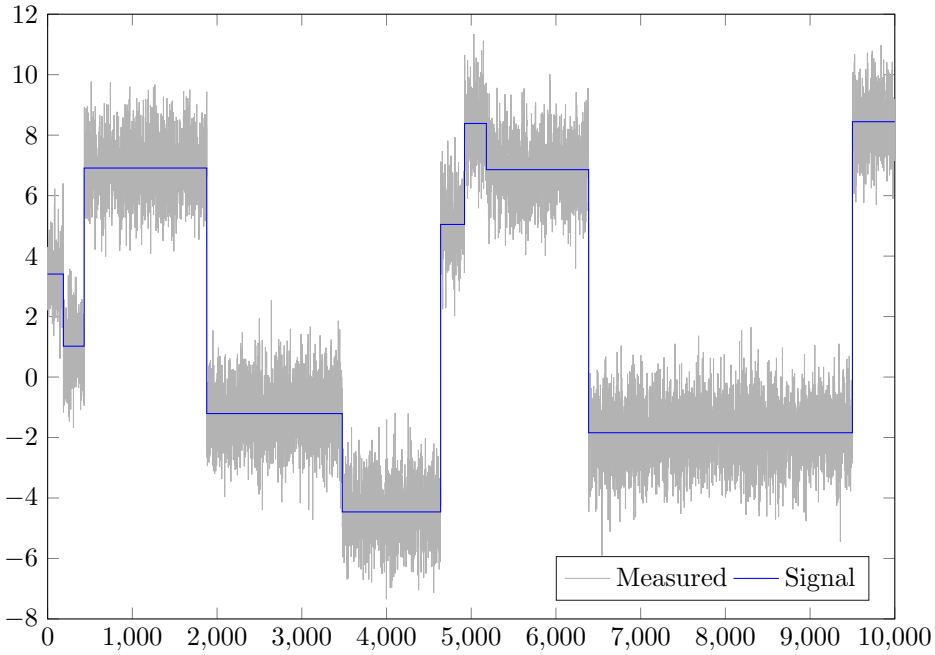


FIGURE 12. The signal piecewise_constant.mat with and without noise

3.6.2. Piecewise Constant

The test signals were generated calling `gen_piecewise_constant.m` (Appendix A.1.1) using the default settings. There are therefore 10 steps and a total of 10,000 samples, corrupted by AGWN, $\sigma = 1$ obtained using `gen_iid_white_noise.m` (Appendix A.1.3). A typical signal developed using this method is shown in Figure 12.

When estimation and denoising are performed as an offline postprocessing step, use of non-causal global optimization algorithms is viable. Figures 39 and 40 quantify performance of the globally minimum ℓ^0 norm path found using VIVA, as well as regularization using the ℓ^1 norm: total variation minimization, a convex optimization approach to regularization presented by Boyd and Vandenberghe (2004, p. 308) as seeking sparsity. The complex optimization is performed using a dual interior-point method implementation by Little and Jones (2010).

VIVA always processes data in order. In offline usage the algorithm runs to completion and reports results based on the entire data set. In this piecewise-constant case without any optional features enabled, retrospective application of VIVA becomes identical to the PELT method of Killick et al. (2012).

The Little and Jones implementation efficiently finds the optimal solution to the ℓ^1 norm regularization problem. Figures 13 and 41 show that bicriterion regularization using the ℓ^0 norm outperforms total variation minimization on all performance metrics, and does so with a simpler (more sparse) estimate. This is because the ℓ^1 norm penalizes step size, biasing the estimate, and creates sparsity as a synergistic effect. In contrast, ℓ^0 norm regularization directly rewards sparsity.

The maximum absolute vertical error metric is particularly unfair to low sparsity estimators, because of the step changes in the input signal. An estimate that misses the step timing by even one sample has a vertical error equal to the entire step in the input signal, despite a low Mahalanobis distance. Meanwhile, a smoothed estimate tends to cross near the midpoint of the step, yielding half as much maximum error. The RMS error also emphasizes occasional large errors.

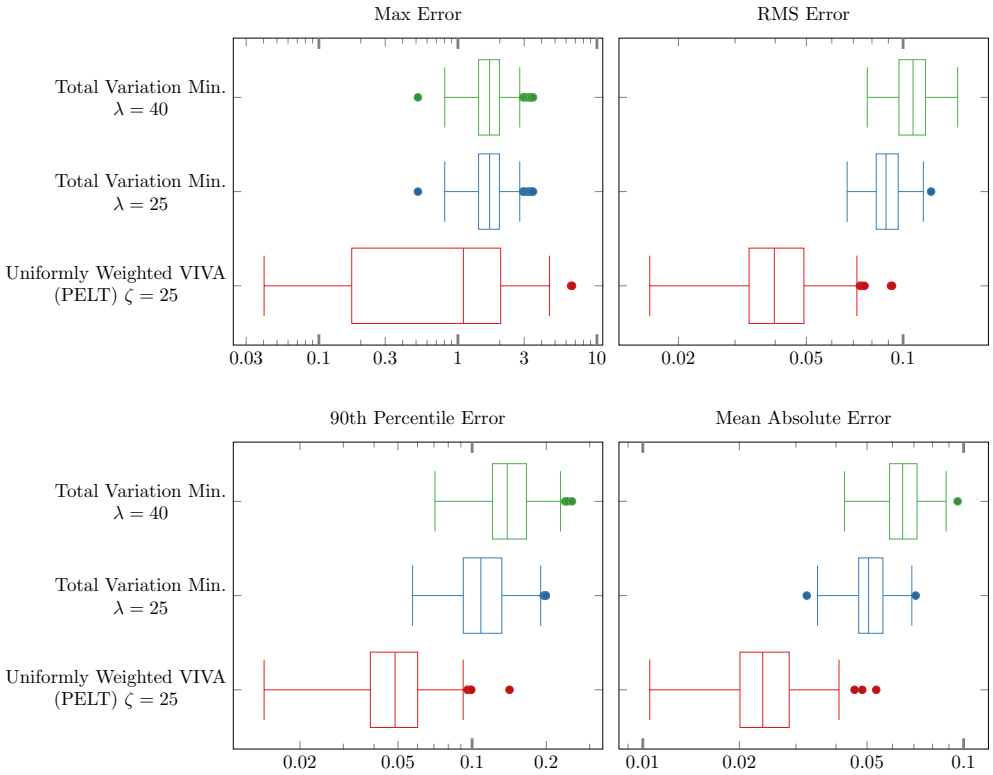


FIGURE 13. Performance of offline estimators evaluated using piecewise constant signal (strongly white noise), N = 240

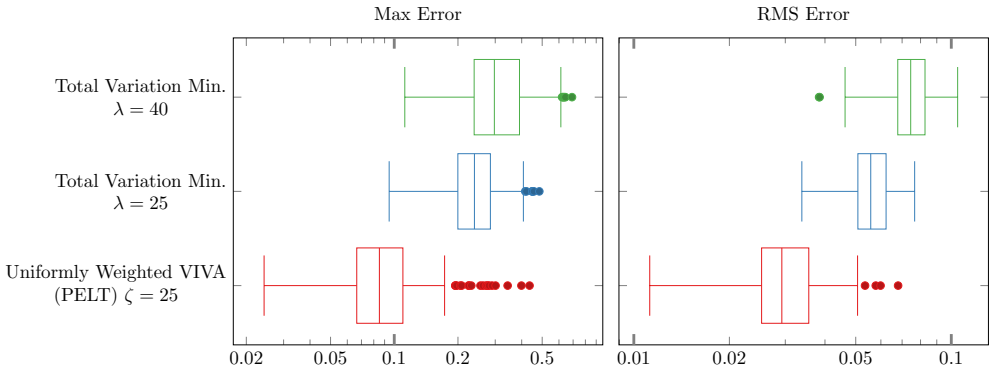


FIGURE 14. Performance in stable regions (transition bands excluded) of offline estimators evaluated using piecewise constant signal (strongly white noise), N = 240

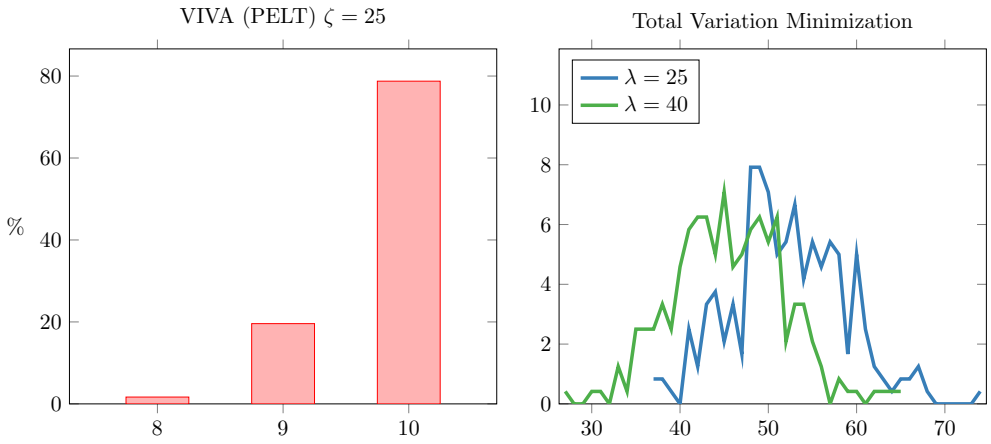


FIGURE 15. Sparsity achieved by pruned exact search compared to total variation minimization estimators evaluated using piecewise constant signal (strongly white noise), N = 240

The 90th percentile and mean absolute vertical error plots tell a different story. Yes, the sparse estimate produced by VIVA may momentarily have a very large error, right near a transition. But these large errors are very few, and PELT-VIVA estimation error away from transitions (Figure 14) is far superior to that yielded by other methods.

3.6.3. Piecewise Linear

We return to the comparison of the final estimate made by VIVA with both weighting schemes after all data are considered, to results obtained using the convex total variance minimization method. However, while the direct application of total variance minimization to the signal does mitigate the white noise as seen in Figure 16 as "Total Variation Min on Value", it produces a stairstep estimate; this results in an extremely erratic rate estimate (Figure 18).

In order to get a more accurate rate signal, a different approach is taken. The rate signal is calculated using a discrete derivative and then smoothed using total variation minimization. This technique is labeled as "Total Variation Min on Rate" in Figures 16 and 18. "TVM first Value, then Rate" will be discussed in conjunction with burst noise.

As previously seen with the piecewise-constant test signals, the rate estimates generated by VIVA have maximum error similar to results from ℓ^1 norm minimization. But VIVA encounters these large residual errors only transiently, and as a direct result of slight errors in recovery of transition timing. Away from transitions, VIVA has a large advantage.

The piecewise-linear VIVA algorithms apply pruning heuristics which sacrifice global optimality for greatly reduced runtime. Residual error levels for "Global Regression" vs "Paired Segment Fit" show that freezing vertexes two (detected) segments in the past has virtually no effect on the residual error. However, between "Original Timing" and "AW-VIVA Timing", some performance degradation is apparent. Although a factor of 500 increase in maximum error seems excessive, it should be noted that the "Original Timing" does not always correspond to the minimizer of the regularization problem due to the added noise. The 30x increase in RMS error and 2x increase in 90th percentile error are more descriptive of the actual cost of the aggressive pruning heuristic.

For clarity, the error plots show only the best instances of each application of total variation minimization. The sparsity chart (Figure 20) shows all tested parameter values. Recall that the signals have all been generated with 10 segments.

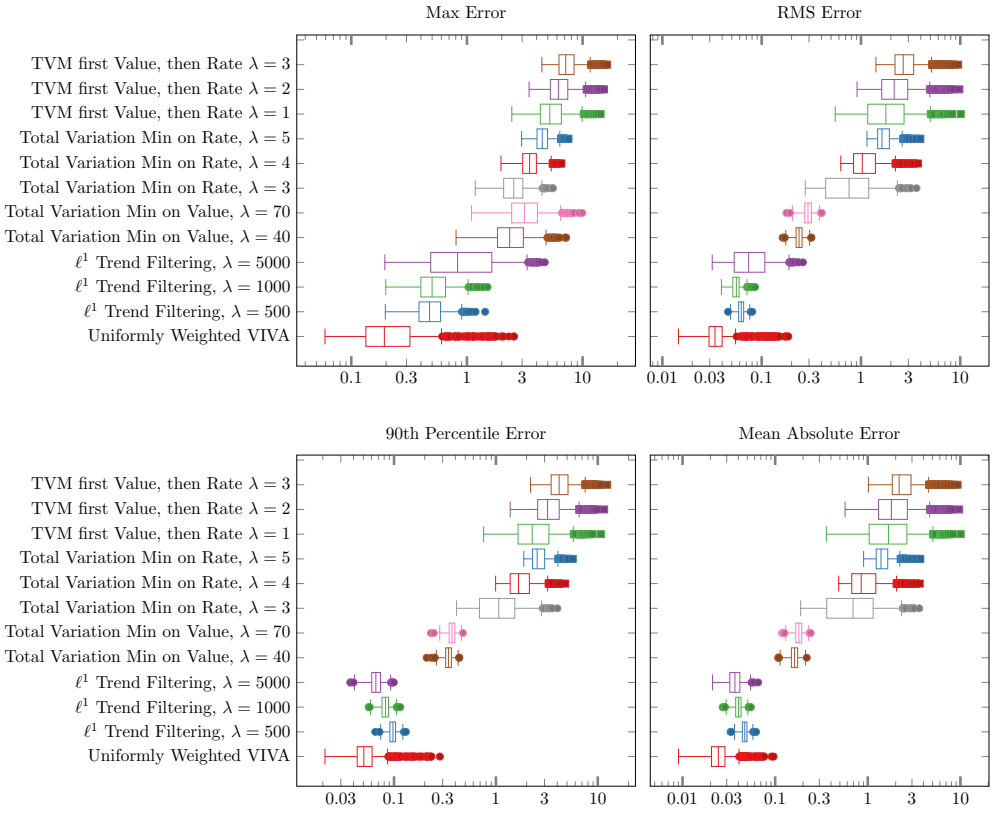


FIGURE 16. Performance of offline estimators evaluated using piecewise linear signal (strongly white noise), N = 500, analysis of value estimate

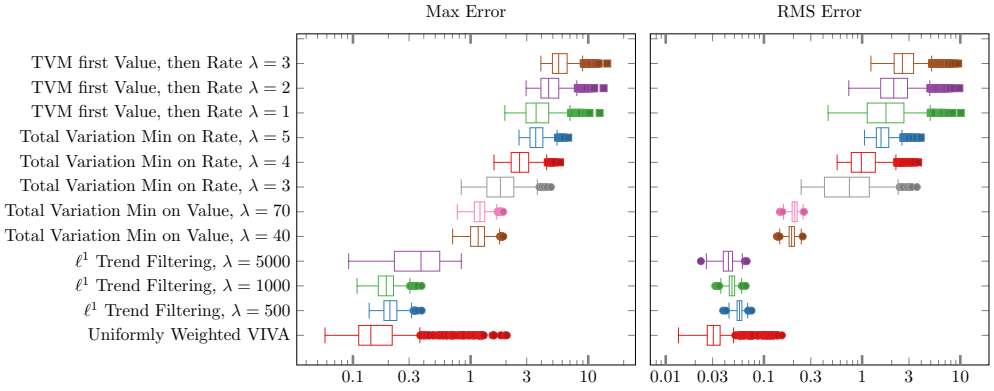


FIGURE 17. Performance in stable regions (transition bands excluded) of offline estimators evaluated using piecewise linear signal (strongly white noise), N = 500, analysis of value estimate

These data suggest that for the strongly white (i.i.d.) Gaussian noise case, chained TVM, performed on value and followed by a derivative and second instance of TVM, is nearly equivalent to TVM performed directly on the derivative of the input.

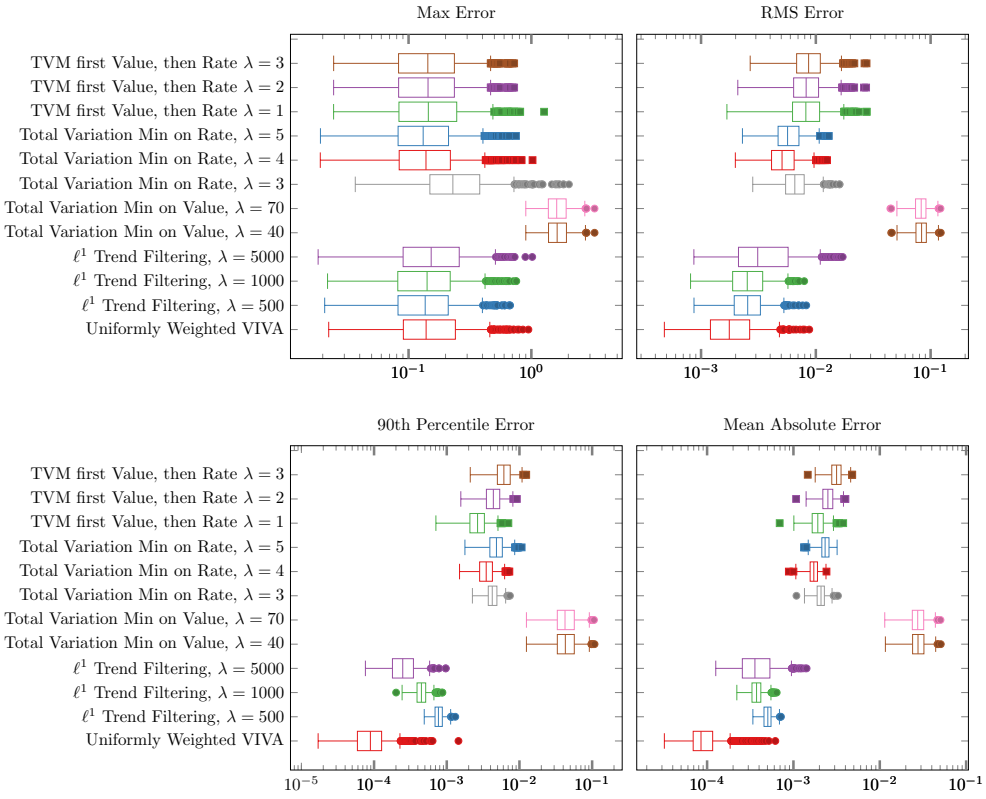


FIGURE 18. Performance of offline estimators evaluated using piecewise linear signal (strongly white noise), N = 500, analysis of rate estimate

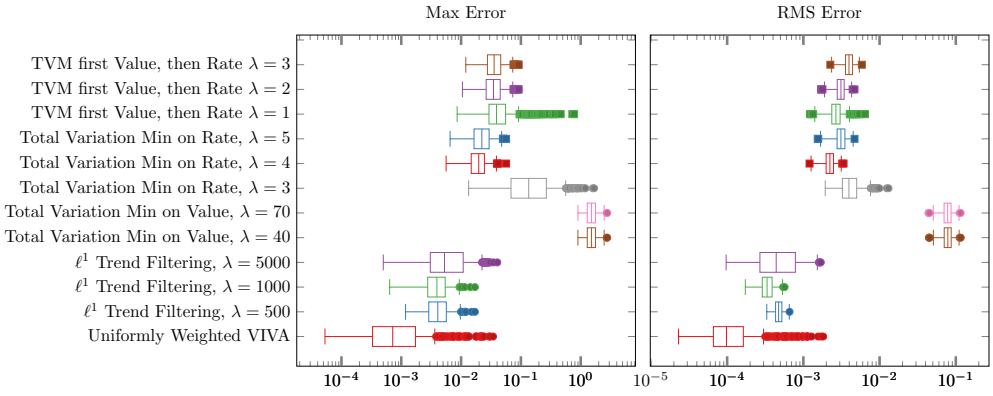


FIGURE 19. Performance in stable regions (transition bands excluded) of offline estimators evaluated using piecewise constant signal (strongly white noise), N = 500, analysis of rate estimate

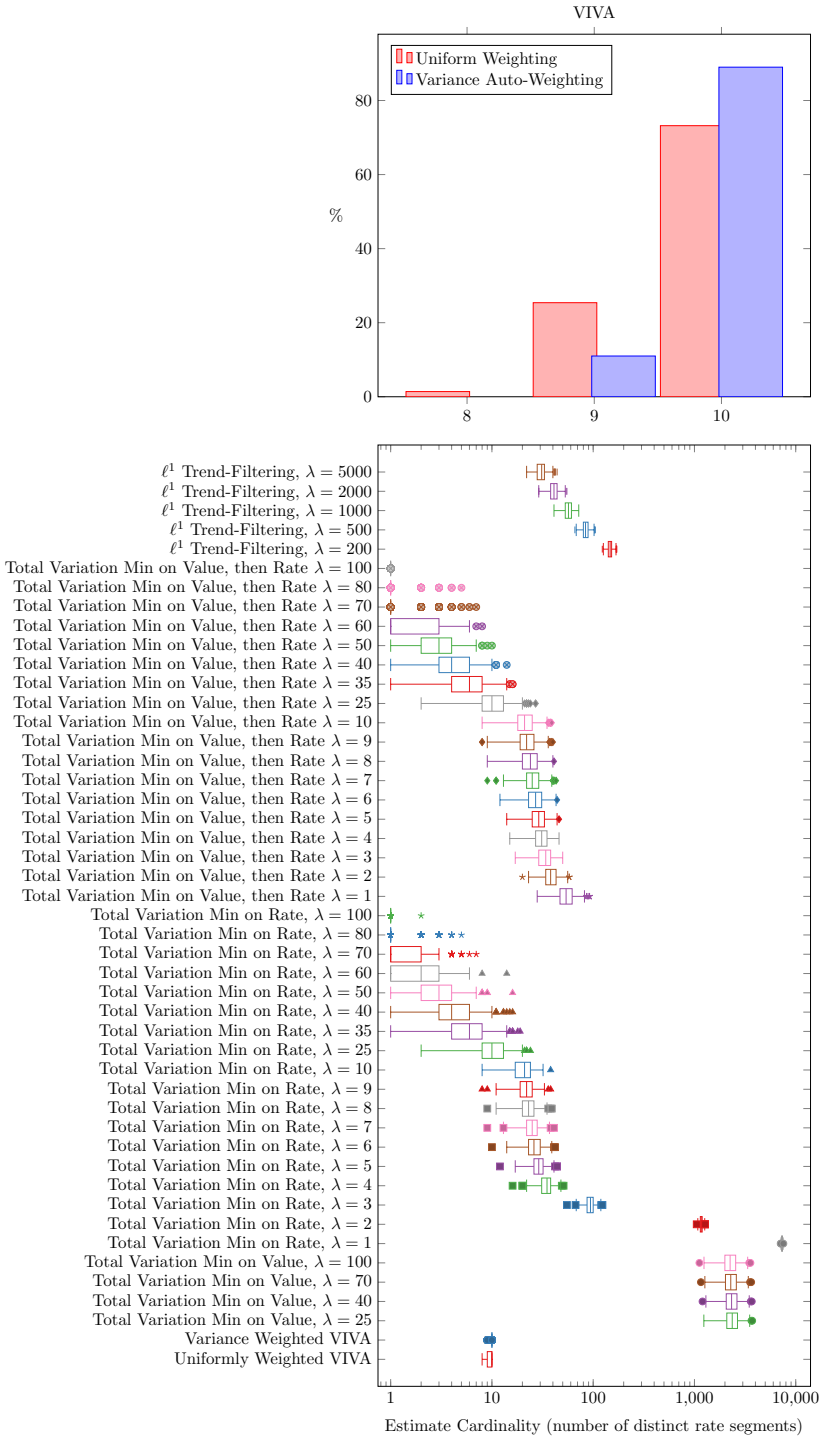


FIGURE 20. Comparison of Sparsity Achieved by all offline methods with piecewise linear signal (strongly white noise), N=500

CHAPTER 4

Confidence Weighting

In many systems, noise characteristics are not stationary. Radio devices operating in shared unlicensed spectrum transmit intermittently. Physical vibrations coupled into load cells from their mounting brackets respond to contact from users. Vehicle-mounted measurement equipment is subjected to widely varying road conditions and radio paths. Detectors operating under these conditions often benefit from adaptive noise rejection.

In this chapter, the bicriterion cost function is shown to correspond to stationary Gaussian noise. A time-varying weight term is introduced to address non-stationary noise, and a weighting formula is proposed. Monte-Carlo analysis shows that inclusion of the weighting term results in significantly lower estimation error.

4.1. Measurement Variance

Recall that the choice of integral-square error in the bicriterion cost function (Equation 1.2) was motivated by negative log-likelihood of i.i.d. Gaussian noise. The Gaussian probability distribution function is

$$\begin{aligned} f(x,\mu,\sigma) &= \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \\ -\log f(x,\mu,\sigma) &= \log(\sigma) + \log(\sqrt{2\pi}) + \frac{(x-\mu)^2}{2\sigma^2} \end{aligned} \tag{4.1}$$

The second term is a constant; the first is a property of external influences on the measurement process. Therefore the entire likelihood is maximized when the third term of the log likelihood is minimized.

$$\underset{x'}{\text{minimize}} \sum_k \frac{(y_k - x'_k)^2}{2\sigma^2} \tag{4.2}$$

If the noise variance is time-invariant, the denominator may be factored out of the sum and blended into the bicriterion balance ζ. (This is a case of special interest for Kalman filters as well, where time-invariant process and measurement noise covariance matrices cause convergence to a time-invariant Kalman gain.)

Otherwise, the $\frac{1}{2}$ may be factored out but the noise variance should be preserved in a weighting term, serving as a measure of the confidence of the system in any single observation:

$$\underset{x'}{\text{minimize}} \sum_k c_k (y_k - x'_k)^2 \tag{4.3}$$

Even if the noise variance is not specifically known, it may be useful to derive weighting coefficients from quantitative indicators of noise. For example, load cells are sensitive to motion, which data from a co-located accelerometer would indicate. Directional communication systems (radio or optical) might measure ambient power levels received off-axis as an indicator of background noise.

4.1.1. Inclusion of Weighting Term in Bicriterion Regularization

The weighting coefficients are easily integrated into the optimization problems 1.7 and 1.8 by allowing the cost function to become the quadratic form of a diagonal matrix:

$$\begin{array}{lrcccl} \text{minimize} & (\mathbf{x}'-\mathbf{y})^T \operatorname{diag}(\mathbf{c})\,(\mathbf{x}'-\mathbf{y}) + \zeta \mathbf{1}^T\mathbf{b}' & & & & \\ \text{subject to} & -x'_k + x'_{k+1} - Mb'_k & \leq & 0 & & \forall k \\ & x'_k - x'_{k+1} - Mb'_k & \leq & 0 & & \forall k \\ & b'_k \in \{\,0,\,1\,\} & & & & \forall k \end{array} \tag{4.4}$$

or for continuous piecewise-linear,

(4.5)

$$\begin{array}{lrcccl} \text{minimize} & (\mathbf{x}'-\mathbf{y})^T \operatorname{diag}(\mathbf{c})\,(\mathbf{x}'-\mathbf{y}) + \zeta \mathbf{1}^T\mathbf{b}' & & & & \\ \text{subject to} & (t_i - t_{i+1})x'_{k-1} + (t_{i+1} - t_{i-1})x'_k + (t_{i-1} - t_i)x'_{k+1} - Mb'_k & \leq & 0 & & \forall k \\ & (t_{i+1} - t_i)x'_{k-1} + (t_{i-1} - t_{i+1})x'_k + (t_i - t_{i-1})x'_{k+1} - Mb'_k & \leq & 0 & & \forall k \\ & b'_k \in \{\,0,\,1\,\} & & & & \forall k \end{array}$$

In practice it is never necessary to form the matrix $\operatorname{diag}(\mathbf{w})$. Rather, the system of equations defining stationary points is rewritten with the weights included, yielding a solution of the same form as the original system of equations; only the computation of the state variables changes:

$$N_c(t_A, t_B) = \sum_{i|t_A<t_i\leq t_B} c_i \tag{4.6a}$$

$$T_c(t_A, t_B) = \sum_{i|t_A<t_i\leq t_B} c_i(t_i - t_A) \tag{4.6b}$$

$$T_c^2(t_A, t_B) = \sum_{i|t_A<t_i\leq t_B} c_i(t_i - t_A)^2 \tag{4.6c}$$

$$S_c(t_A, t_B) = \sum_{i|t_A<t_i\leq t_B} c_i y_i \tag{4.6d}$$

$$S_c^2(t_A, t_B) = \sum_{i|t_A<t_i\leq t_B} c_i y_i^2 \tag{4.6e}$$

$$R_c(t_A, t_B) = \sum_{i|t_A<t_i\leq t_B} c_i(t_i - t_A) y_i \tag{4.6f}$$

These remain amenable to recursive computation:

$$N_c(t_A, t_i) = N_c(t_A, t_{i-1}) + c_i \tag{4.7a}$$

$$T_c(t_A, t_i) = T_c(t_A, t_{i-1}) + c_i(t_i - t_A) \tag{4.7b}$$

$$T_c^2(t_A, t_i) = T_c^2(t_A, t_{i-1}) + c_i(t_i - t_A)^2 \tag{4.7c}$$

$$S_c(t_A, t_i) = S_c(t_A, t_{i-1}) + c_i y_i \tag{4.7d}$$

$$S_c^2(t_A, t_i) = S_c^2(t_A, t_{i-1}) + c_i y_i^2 \tag{4.7e}$$

$$R_c(t_A, t_i) = R_c(t_A, t_{i-1}) + c_i(t_i - t_A) y_i \tag{4.7f}$$

The full derivation is provided in Appendix B.

4.1.2. Automatic Weighting by Variance Estimation

If no independent indicator of noise variance is available, then the observed signal itself may be the best estimator. One possibility is to use the local variance of the measurement sequence. For resilience to time-varying noise power, VIVA implements the following weighting formula:

Using a sliding window with support $2s + 1$

$$c_k^{-1} = 1 + \frac{1}{2s+1}\left(\sum_{i=k-s}^{k+s} y_i^2\right) - \left(\frac{1}{2s+1}\sum_{i=k-s}^{k+s} y_i\right)^2 \tag{4.8a}$$

Although the window variance is influenced by the trend in the true data (for segment models other than piecewise-constant), if any timing errors exist in the observations, then the window variance has the desirable effect of counteracting the difference between integral-square residual error and Mahalanobis distance.

Window variance also increases in the neighborhood of changepoints, where it helps to accommodate parameter changes which are not precisely instantaneous and may occur over several measurements.

4.2. Burst Noise and Denoising Performance

To simulate conditions where individual interfering symbols have shorter duration that the sampling interval, but arrive as pulse trains affecting a sequence of measurements, noise is generated such that samples are individually Gaussian and statistically uncorrelated, but not statistically independent, nor stationary. Each block of 10 samples has

$$w \sim \begin{cases} \mathcal{N}(0,1) & \text{with probability } 95\,\% \\ \mathcal{N}(0,16^2) & \text{with probability } 5\,\% \end{cases} \tag{4.9}$$

The result is white noise in the weak sense, exemplified by Figure 21. A script for generating burst noise is included in Appendix A.1.4.

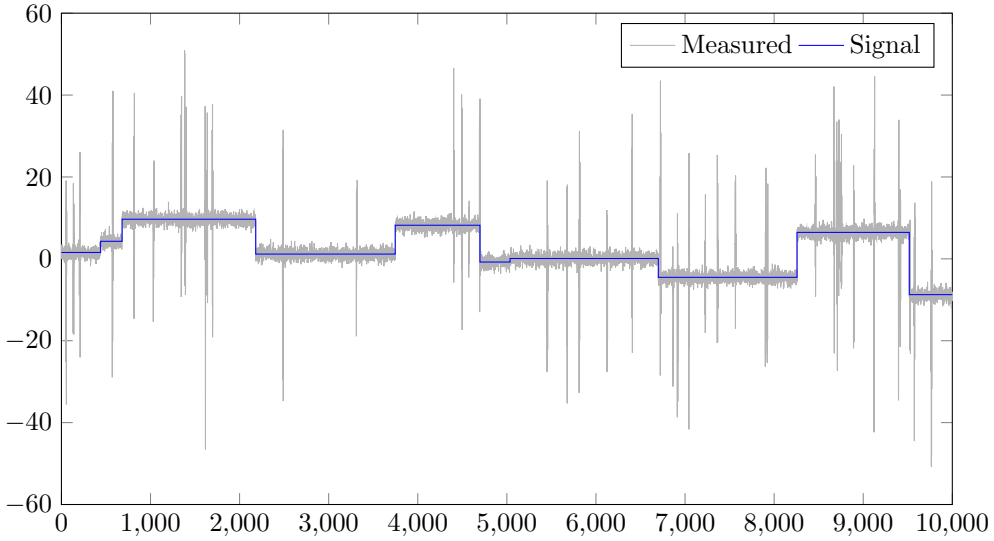


FIGURE 21. The signal piecewise_constant_bursty.mat with and without noise

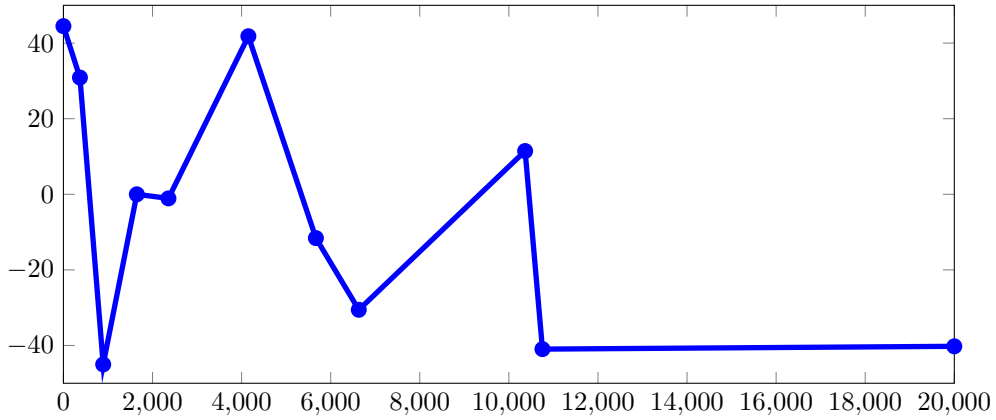


FIGURE 22. "Transmitted" Signal

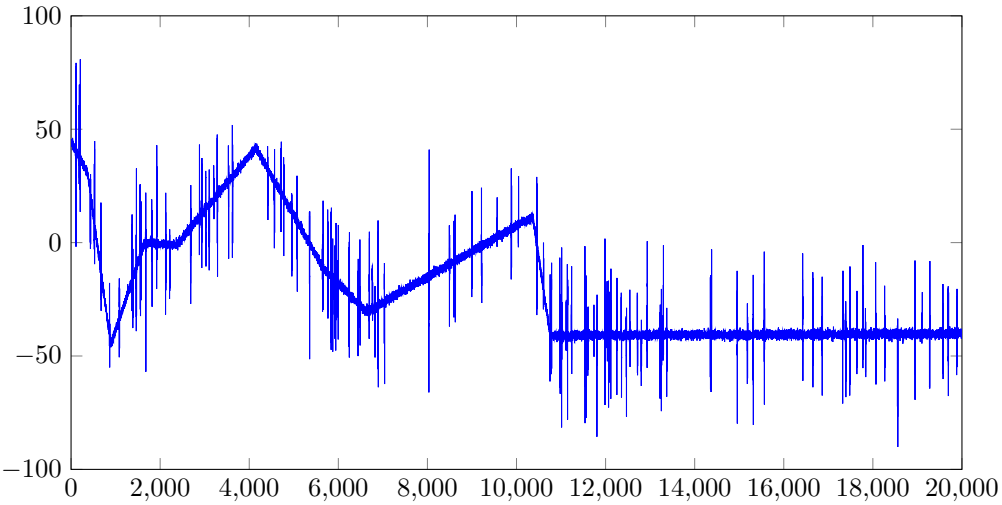


FIGURE 23. "Received" Signal corrupted by burst noise

Burst noise is not specifically associated with disjoint signal models. To visualize the effect on polyline signals, the "transmitted" signal from `piecewise_linear.mat` is again considered, but instead of strongly white noise, burst noise has been generated using `gen_burst_white_noise.m`. Figures 22 and 23 show the result.

This test represents the most realistic simulation of a load cell signal performed. The piecewise-linear signal approximates the volume moved by an IV pump, and the burst noise represents motion of the IV pole following accidental physical contact, while the noise floor includes the aggregation of many vibration sources, EM radiation picked up in the signal cables, and thermal noise in the acquisition circuit. Figure 23 shows an example synthesized signal.

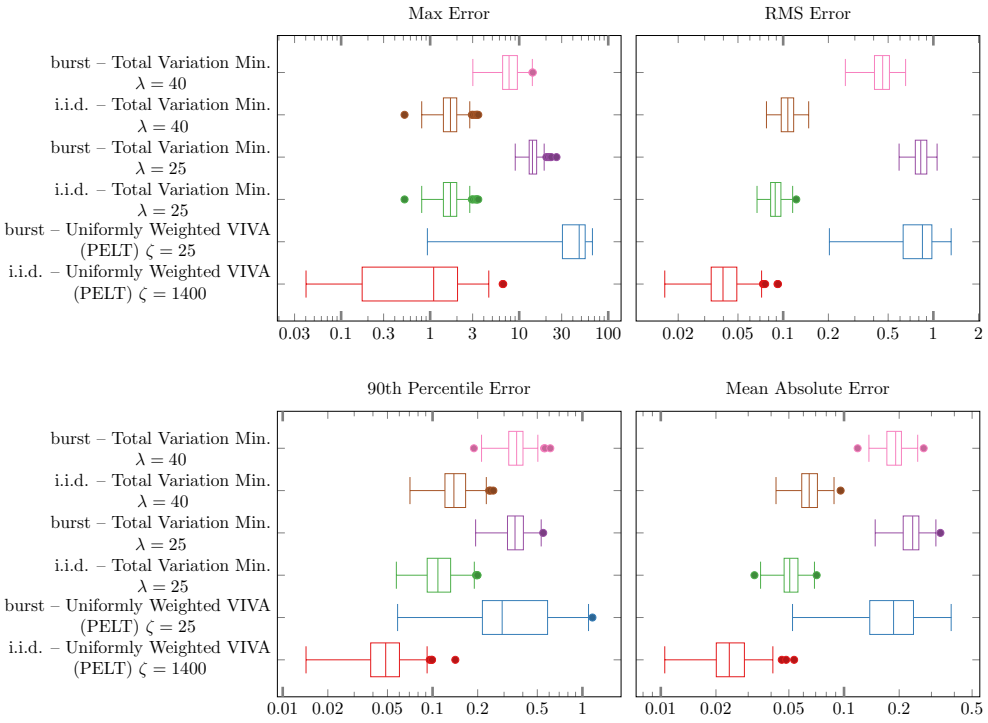


FIGURE 24. Performance degradation of offline estimators when burst noise is introduced into piecewise constant signal

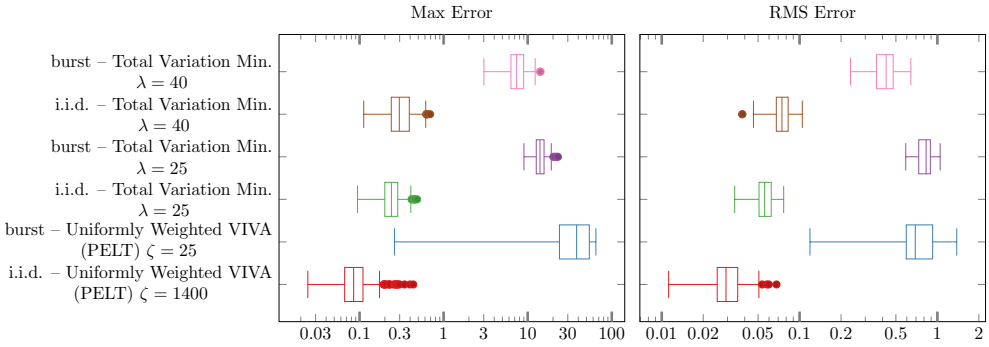


FIGURE 25. Performance degradation in stable regions (transition bands excluded) of offline estimators when burst noise is introduced into piecewise constant signal

4.2.1. Unweighted Estimator Performance Degradation due to Burst Noise

The burst noise has nearly 14 times as much the energy as the i.i.d. noise, and the effect of this increase on estimation of piecewise-constant signals is readily seen in Figure 24.

Figure 26 reveals the reason for lackluster performance of PELT and the uniformly weighted VIVA estimator – it is tricked into thinking some of the noise bursts represent actual input steps. Note that bicriterion balance factor ζ was increased to achieve approximately the correct median sparsity, however the range of sparsity has greatly increased; there is no value of ζ that can restore correct operation. (Compare to the effect of confidence weighting, Figure 33) Meanwhile

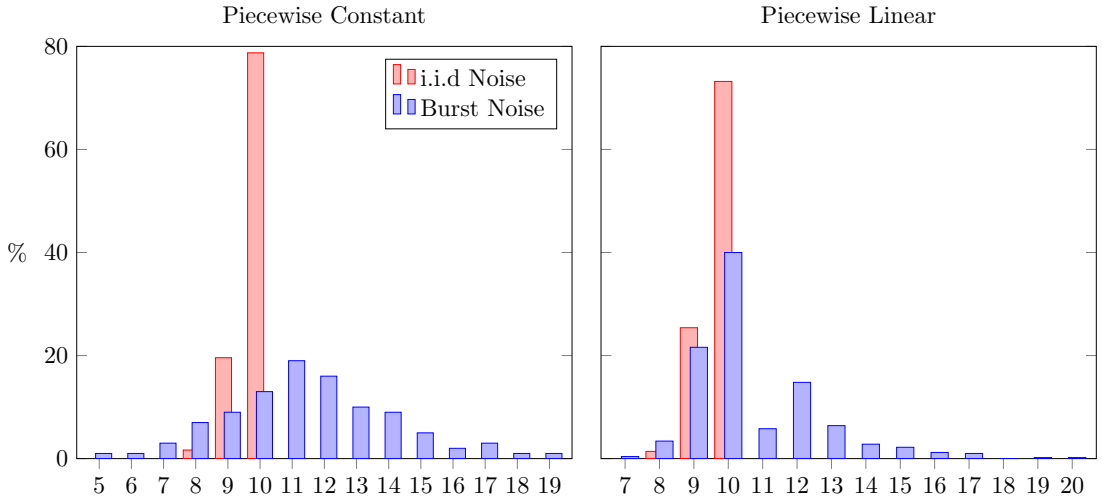


FIGURE 26. Burst noise prevents uniformly weighted VIVA from achieving the correct sparsity pattern for most signals

all configurations of total variation minimization yield increased error commensurate with the RMS increase in noise.

For piecewise linear signals (Figures 27 through 30), the situation is not much different, although uniformly weighted VIVA no longer falls into last place overall, the performance of all estimators has suffered greatly. The range of sparsity (Figure 26) also increased somewhat less for piecewise linear signals than for piecewise constant. Overall, though, static tuning of denoising algorithms is ineffective against time-varying interference.

4.2.2. Variance-Based Weighting Provides Robustness to Burst Noise

While burst noise proved extremely challenging for static uniform weighting, adaptive denoising using the Equation 4.8a weighting formula has no trouble at all. The offline estimator results, Figures 31 and 32, reveal that variance-based weighting achieves the lowest-noise estimate by a wide margin, a result which will be repeated among the online estimators.

Increasing the variation penalty λ of total variation minimization makes it more tolerant of noise. However, Figure 33 shows that excessive numbers of changepoints are still being accepted.

Moving to piecewise-linear signals, Figure 34 contains a remarkable result: The final result from variance auto-weighted VIVA has not only surpassed the other estimates of the value signal by all error measures, it also outperforms the globally optimal solution to the regularization problem. This paradox comes about because the minimum mean-square error fit sought by regularization is not the maximum-likelihood predictor of the original signal under burst noise conditions.

While the AW-VIVA estimate does not outright beat the rate estimate found by global least-squares, it still is the best of the practical estimators. Meanwhile the chained application of TVM generates better rate estimates than single application, but still exhibits average error an order of magnitude worse than VIVA. ℓ^1 trend filtering (Kim et al., 2009) offers significant improvement over TVM by adding a second-order difference term directly to the convex cost function, and TVM might also benefit from pairing with a different lowpass pre-filter. However, in light of the results on piecewise-constant data for which the TVM implementation is designed, it seems impossible that improved ℓ^1 methods should provide an estimate which is simultaneously accurate and sparse.

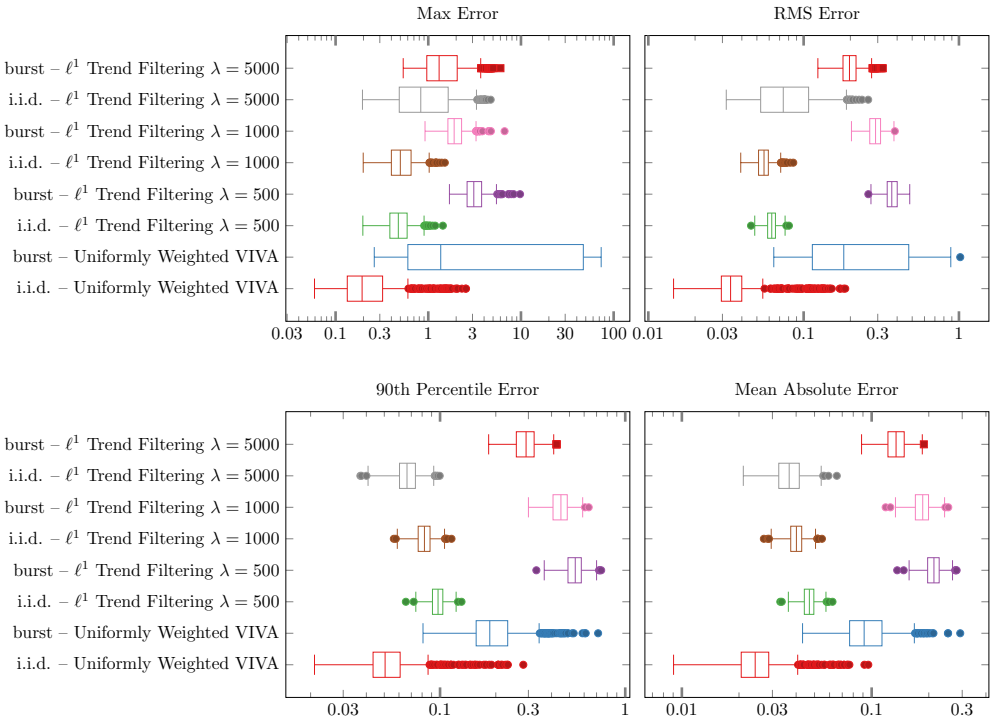


FIGURE 27. Performance degradation of offline value estimates when burst noise is introduced into piecewise linear signal

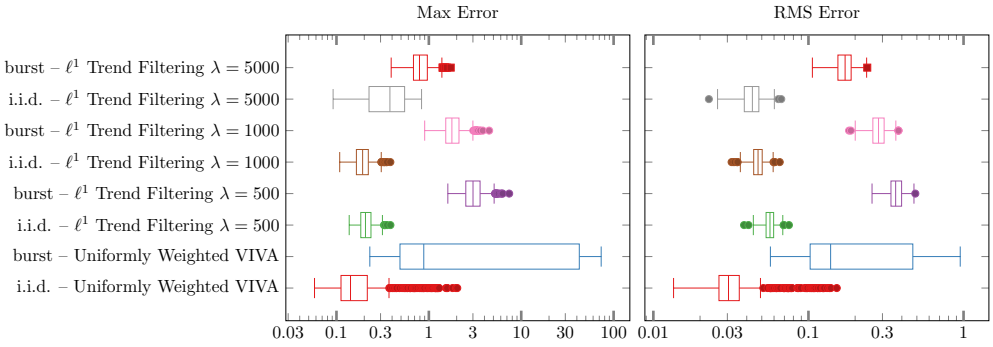


FIGURE 28. Performance degradation in stable regions (transition bands excluded) of offline value estimates when burst noise is introduced into piecewise linear signal

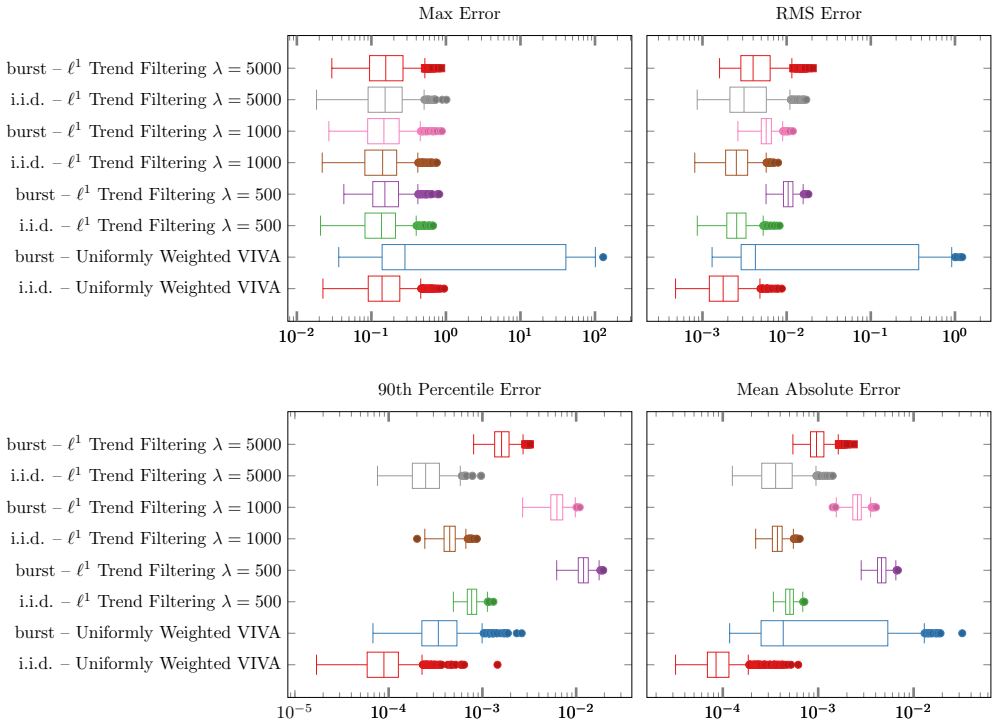


FIGURE 29. Performance degradation of offline rate estimates when burst noise is introduced into piecewise linear signal

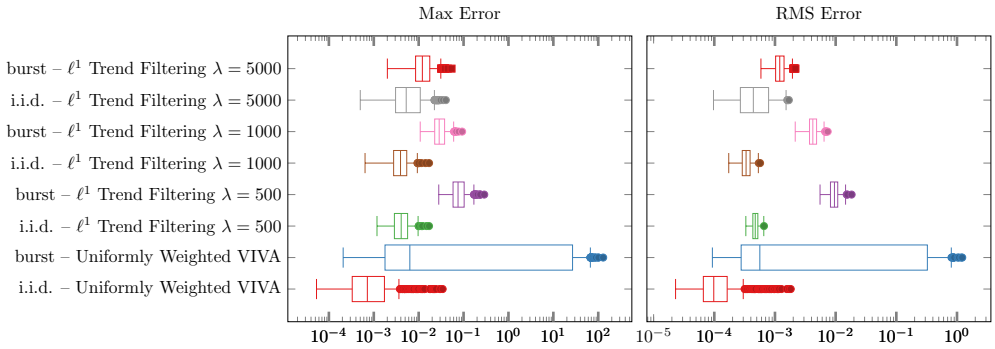


FIGURE 30. Performance degradation in stable regions (transition bands excluded) of offline rate estimates when burst noise is introduced into piecewise linear signal

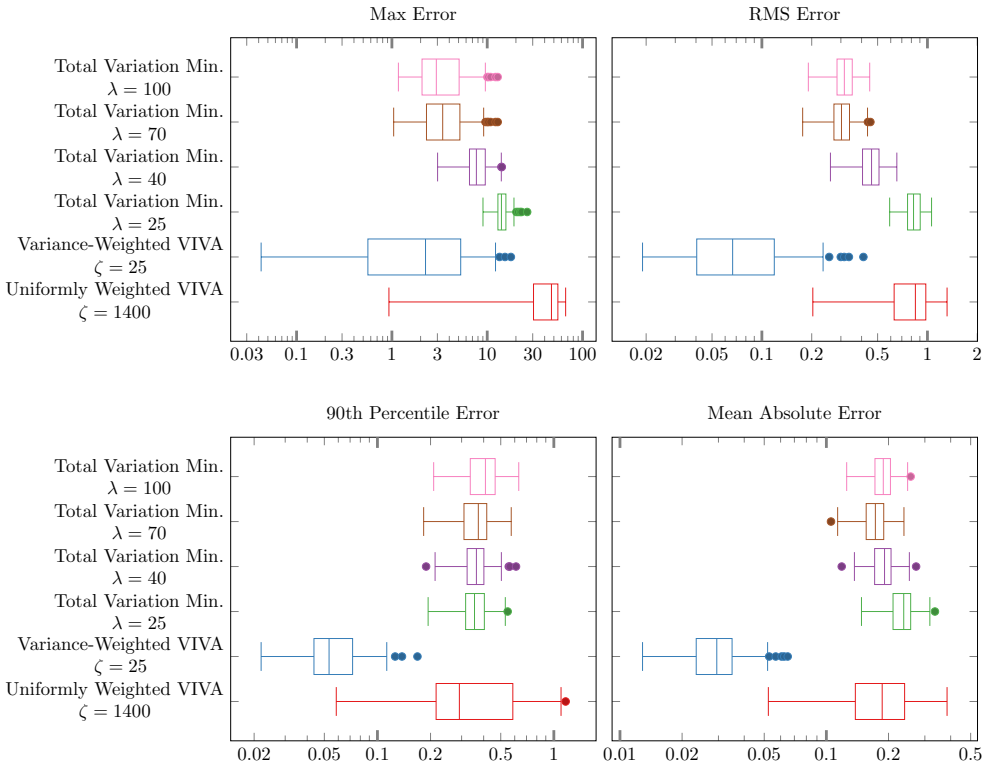


FIGURE 31. Performance of offline estimators evaluated using piecewise constant signal (burst noise), N = 100

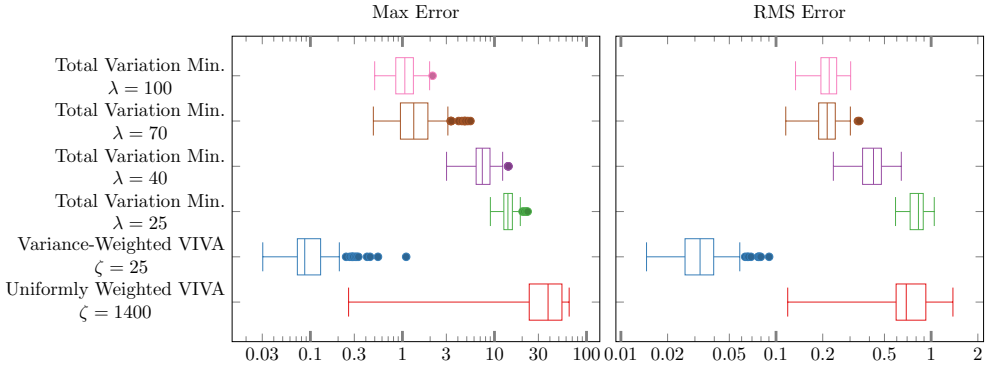


FIGURE 32. Performance in stable regions (transition bands excluded) of offline estimators evaluated using piecewise constant signal (burst noise), N = 100

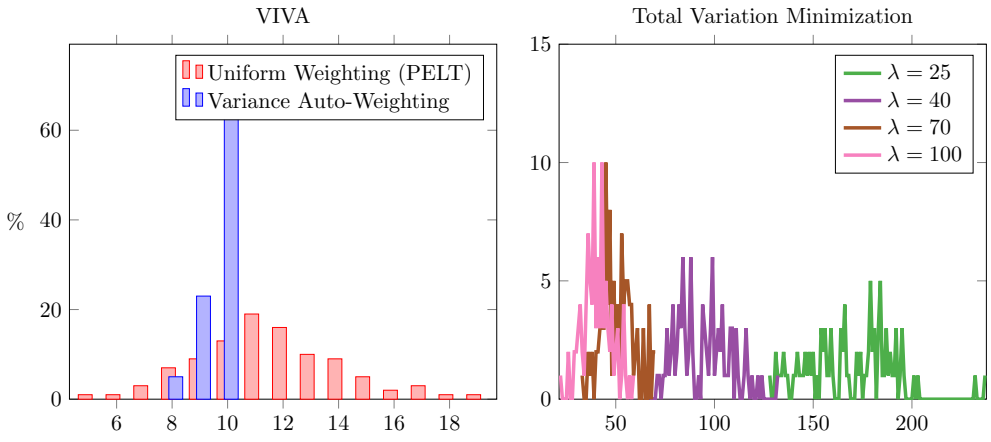


FIGURE 33. Sparsity achieved by offline estimators evaluated using piecewise constant signal (burst noise), N = 100

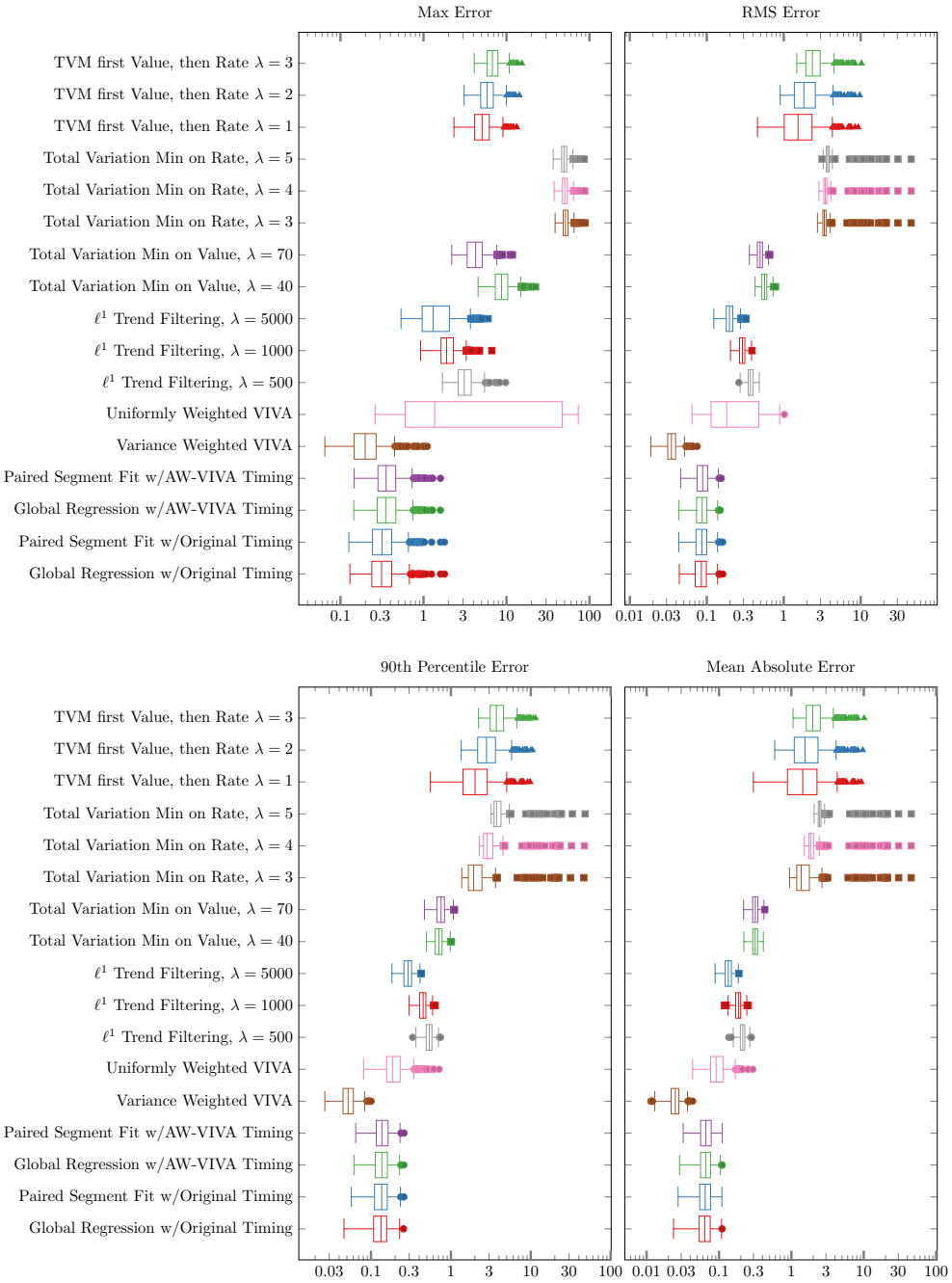


FIGURE 34. Performance of offline estimators evaluated using piecewise linear signal (burst noise), N = 500, analysis of value estimate

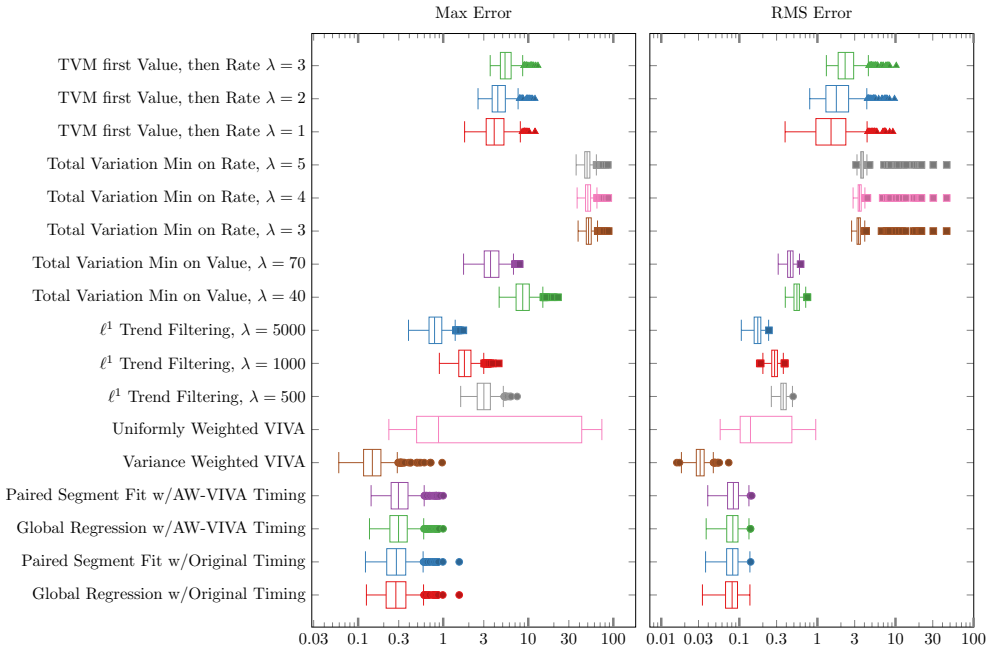


FIGURE 35. Performance in stable regions (transition bands excluded) of offline estimators evaluated using piecewise linear signal (burst noise), N = 500, analysis of value estimate

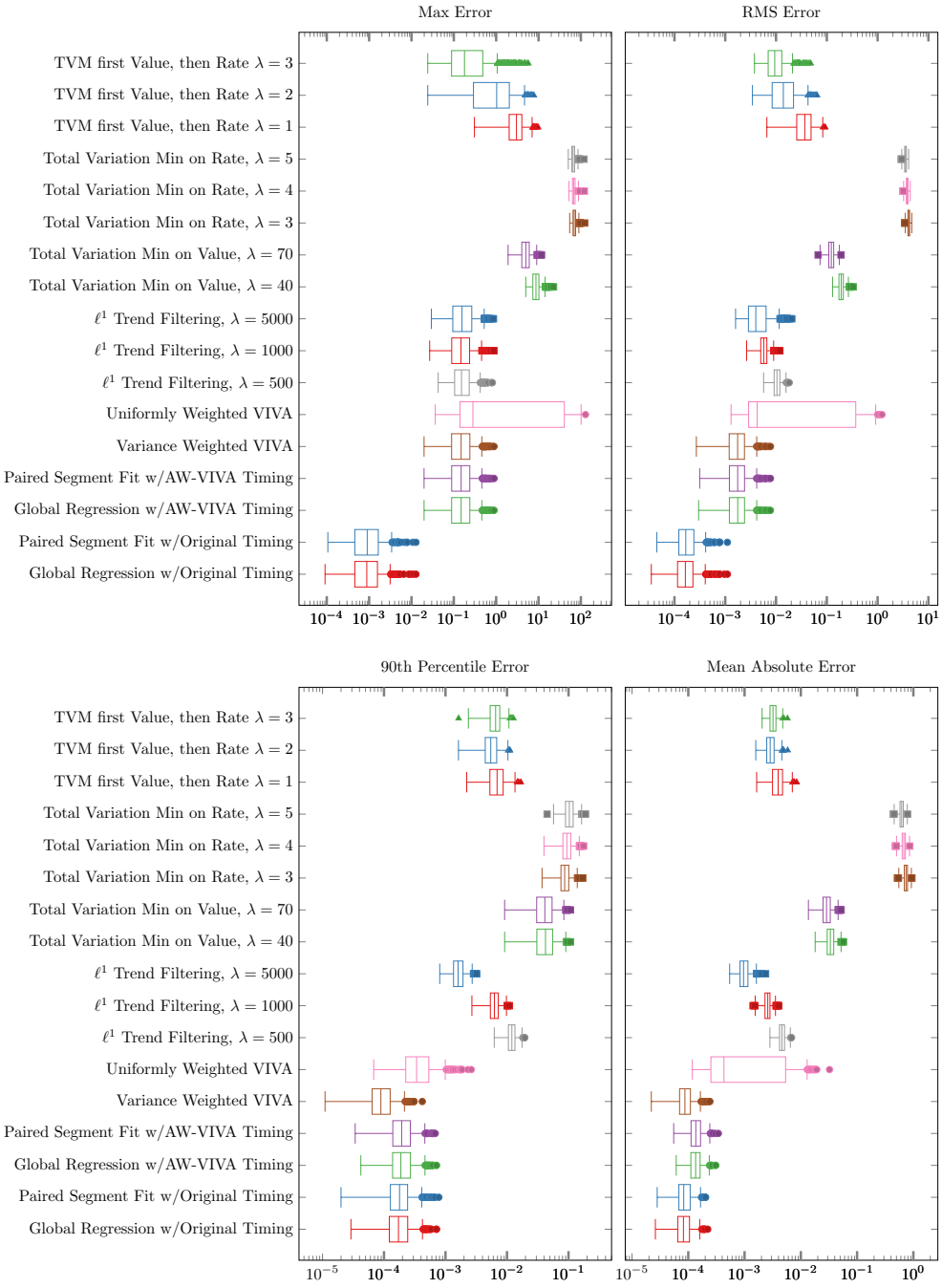


FIGURE 36. Performance of offline estimators evaluated using piecewise linear signal (burst noise), N = 500, analysis of rate estimate

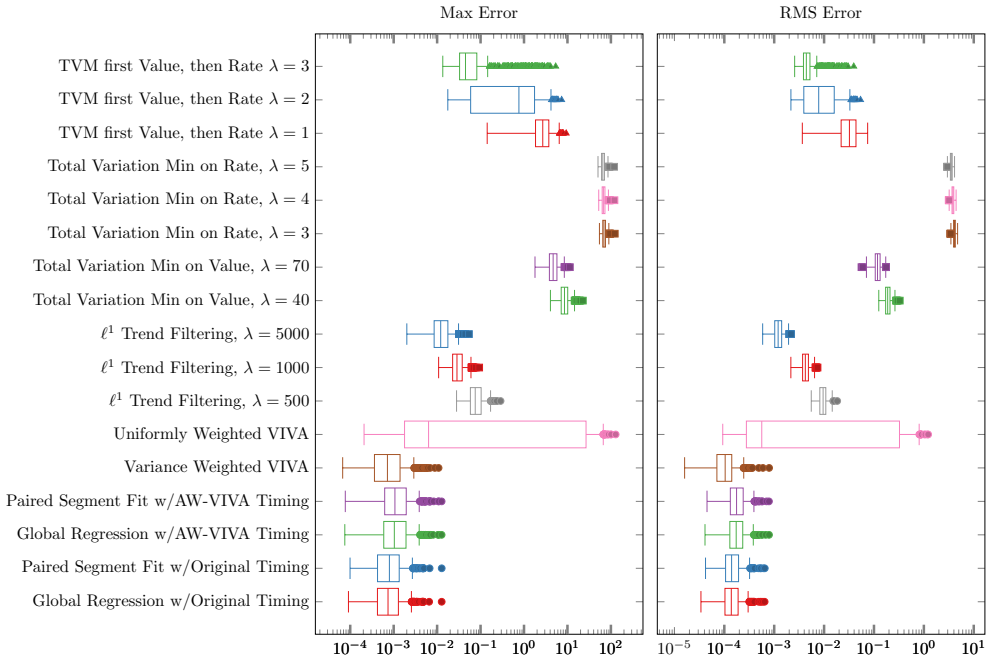


FIGURE 37. Performance in stable regions (transition bands excluded) of offline estimators evaluated using piecewise linear signal (burst noise), N = 500, analysis of rate estimate

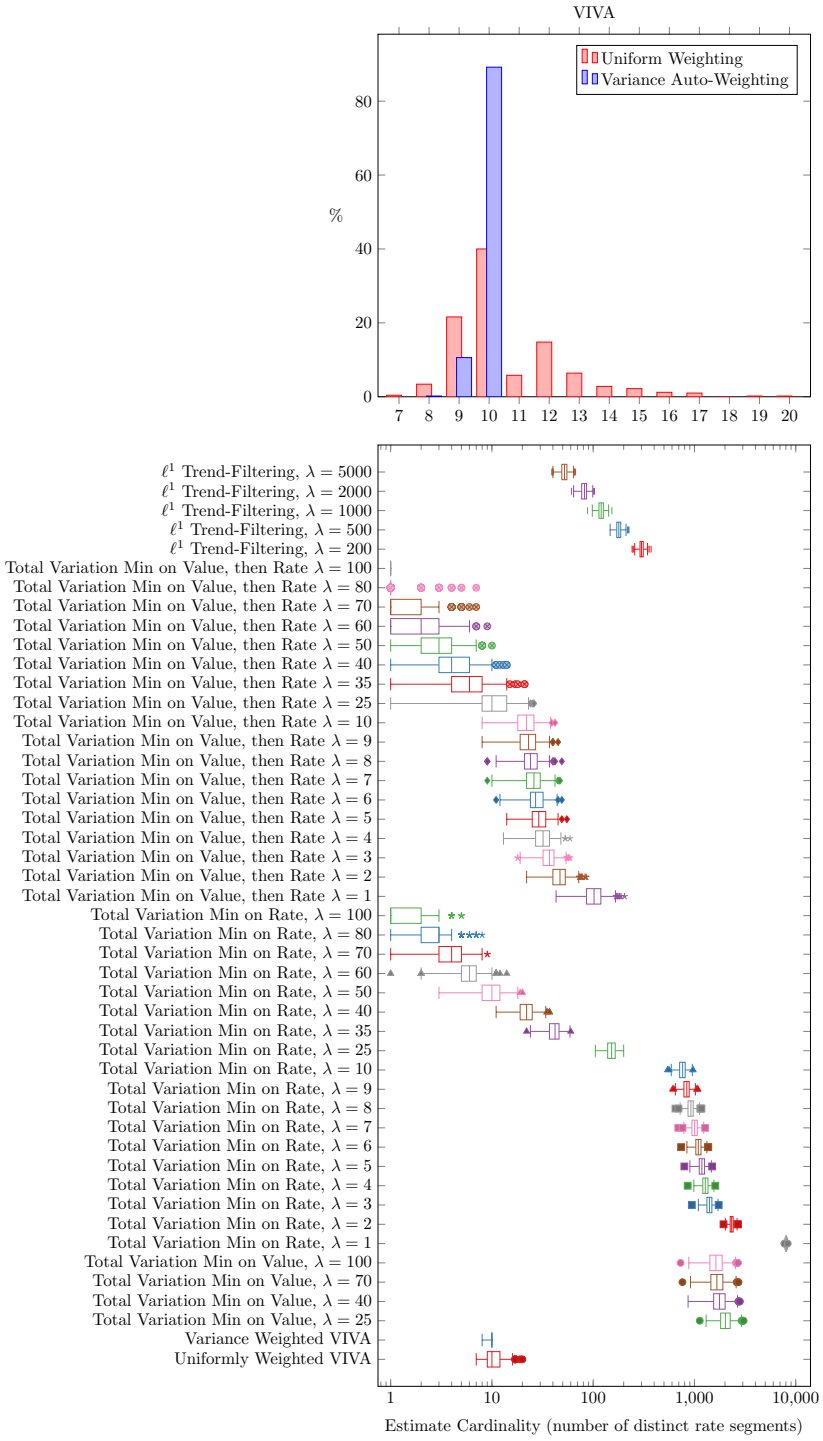


FIGURE 38. Comparison of Sparsity Achieved by all offline methods with piecewise linear signal (burst noise), N=500

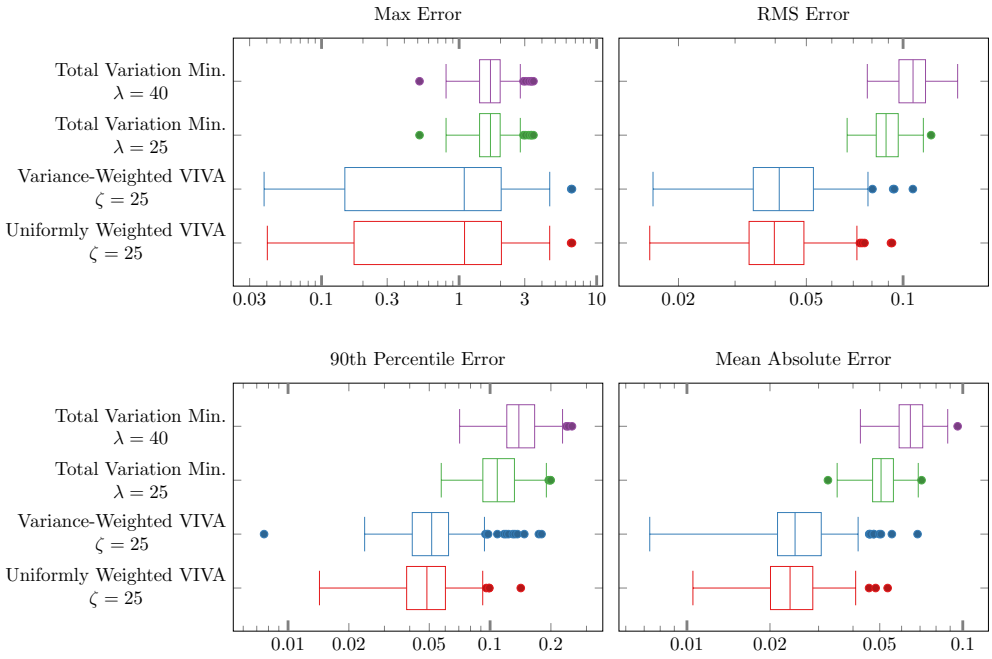


FIGURE 39. Performance of offline estimators evaluated using piecewise constant signal (strongly white noise), N = 240

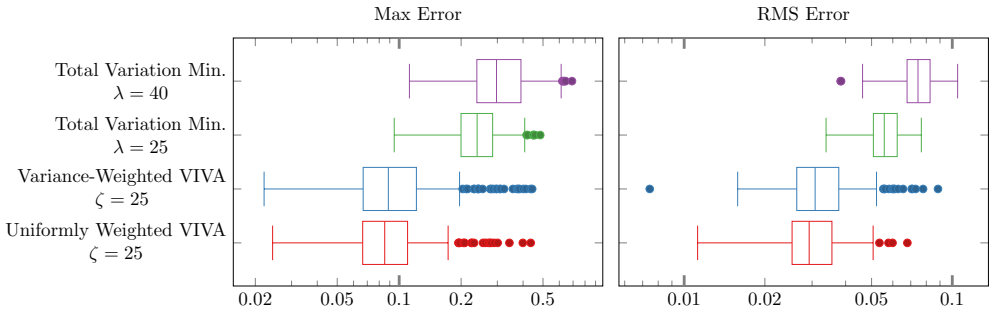


FIGURE 40. Performance in stable regions (transition bands excluded) of offline estimators evaluated using piecewise constant signal (strongly white noise), N = 240

4.2.3. Impact of Variance-Based Weighting on Strictly White Noise

Figures 39 and 41 reveal that for stationary noise, the variance-weighted scheme performs slightly worse that the uniform scheme, because the main source of variance is step changes in the original signal. De-emphasizing transition regions increases the likelihood of selecting the wrong step timing. The effects on online estimation are more pronounced, and will be seen in subsection 5.3.1.

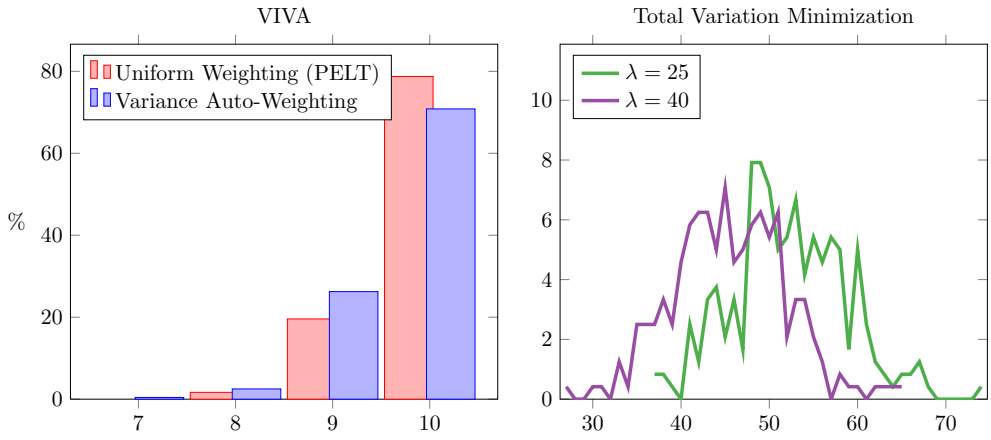


FIGURE 41. Sparsity achieved by offline estimators evaluated using piecewise constant signal (strongly white noise), N = 240

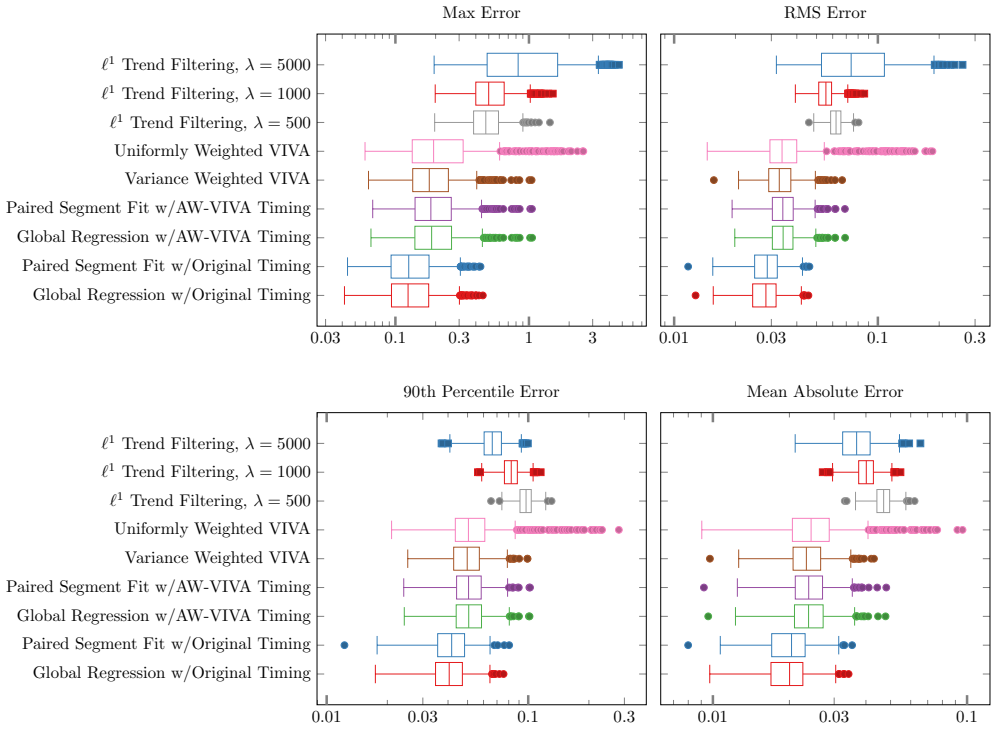


FIGURE 42. Performance of offline estimators evaluated using piecewise linear signal (strongly white noise), N = 500, analysis of value estimate

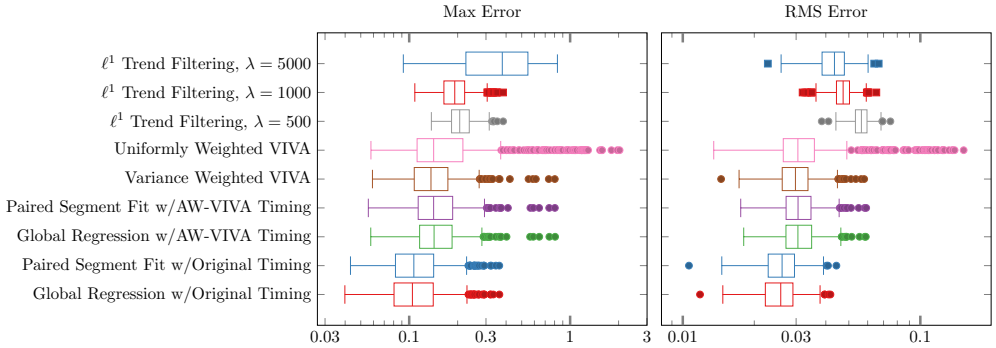


FIGURE 43. Performance in stable regions (transition bands excluded) of offline estimators evaluated using piecewise linear signal (strongly white noise), N = 500, analysis of value estimate

The piecewise linear case is modeled (and generated) as a series of connected line segments. Unlike the piecewise-constant case, this signal is continuous, and has significantly reduced high frequency content. This is beneficial to the various estimators which perform smoothing, at least in terms of residual error in the value estimate. Accurate estimation of the rate signal, which is discontinuous where the segments meet, remains challenging. Rate calculations using numeric differentiation amplify high frequency noise as predicted by the frequency-domain transfer function of the derivative operator: $\mathfrak{F}\left\{\frac{d}{dt}y(t)\right\} = j2\pi f Y(f)$.

The signal '`piecewise_linear.mat`', shown in Figure 46, exemplifies this case. It has been generated using `gen_piecewise_linear.m` (Appendix A.1.2) and corrupted using i.i.d. white

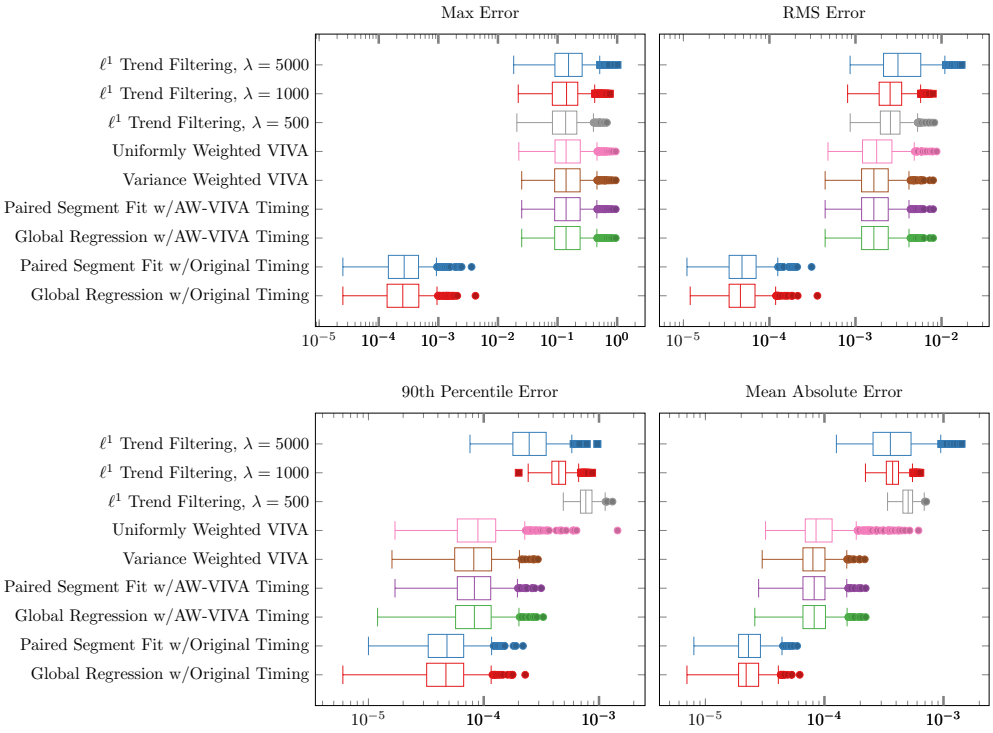


FIGURE 44. Performance of offline estimators evaluated using piecewise linear signal (strongly white noise), N = 500, analysis of rate estimate

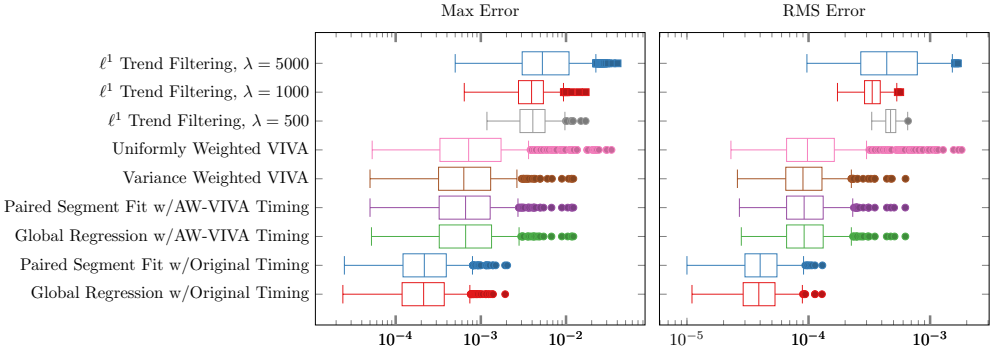


FIGURE 45. Performance in stable regions (transition bands excluded) of offline estimators evaluated using piecewise constant signal (strongly white noise), N = 500, analysis of rate estimate

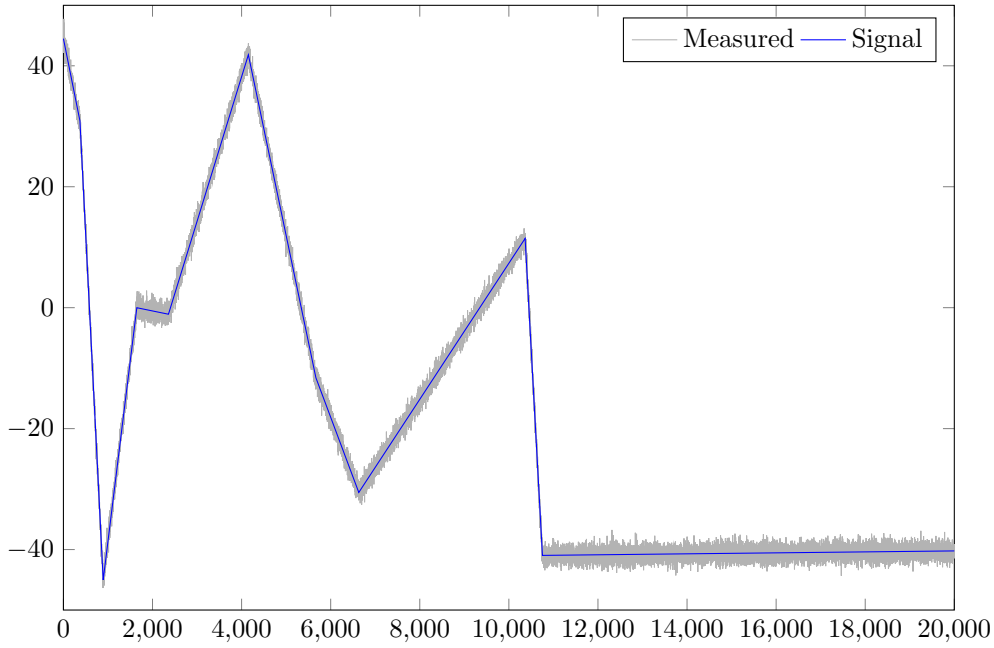


FIGURE 46. The signal `piecewise_linear.mat` with and without noise

Gaussian noise obtained using `gen_iid_white_noise.m` just as in the piecewise constant case. Once again the randomly selected parameters used during signal generation are saved for use in evaluation of various estimates.

Without the high local variance contributed by step changes in the input, variance auto-weighting is no longer devaluing transition regions, and as a result the auto-weighting scheme is no longer disadvantaged relative to the uniformly weighted version. Both VIVA variants benefit from the continuous value signal; without the sudden steps small timing errors no longer equate to large maximum error levels, at least not in the value estimate (Figure 42). However because step changes do exist in the rate signal (Figure 44), VIVA is again subject to large errors, although these are limited to transition regions.

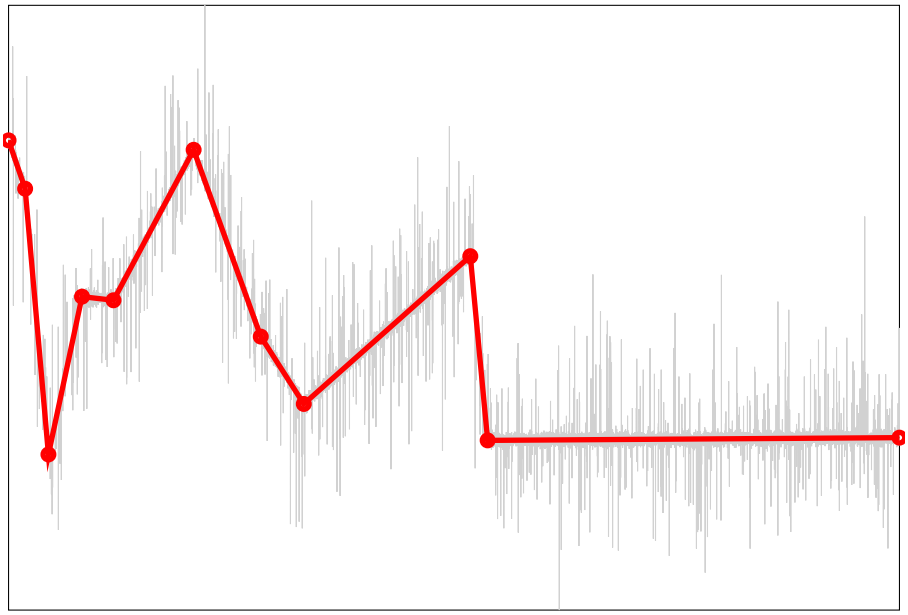


FIGURE 47. Signal corrupted by sporadic noise and recovered using inverse-variance auto-weighting

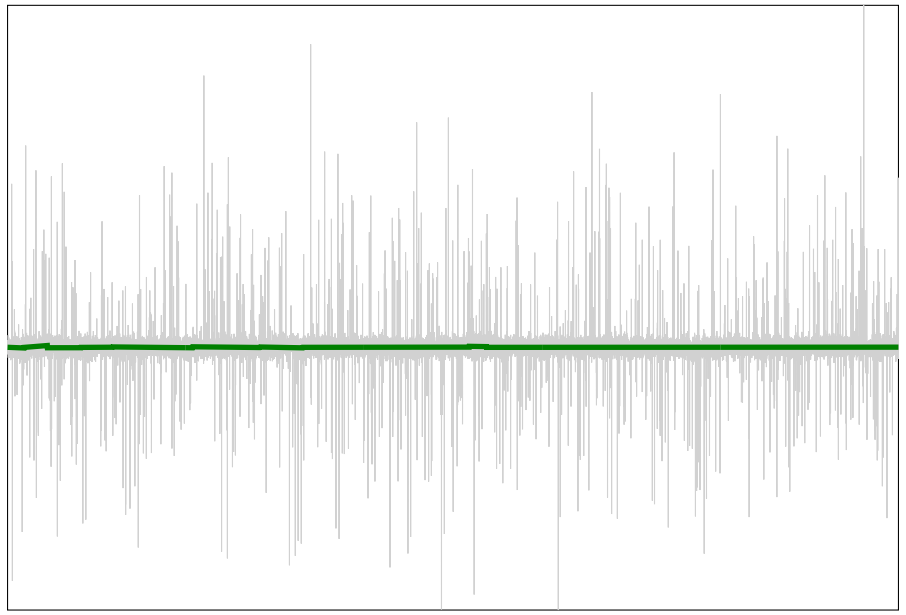


FIGURE 48. Sporadic noise and residual error after recovery using inverse-variance auto-weighting

4.2.4. Spectrum of residual noise

Consider the signal shown in Figure 47, which has been synthesized with "sporadic" noise (generated using `gen_burst_white_noise.m` and a burst length of `1`). Subtracting out the original signal leaves the measurement noise and offline/final residual error following VIVA denoising

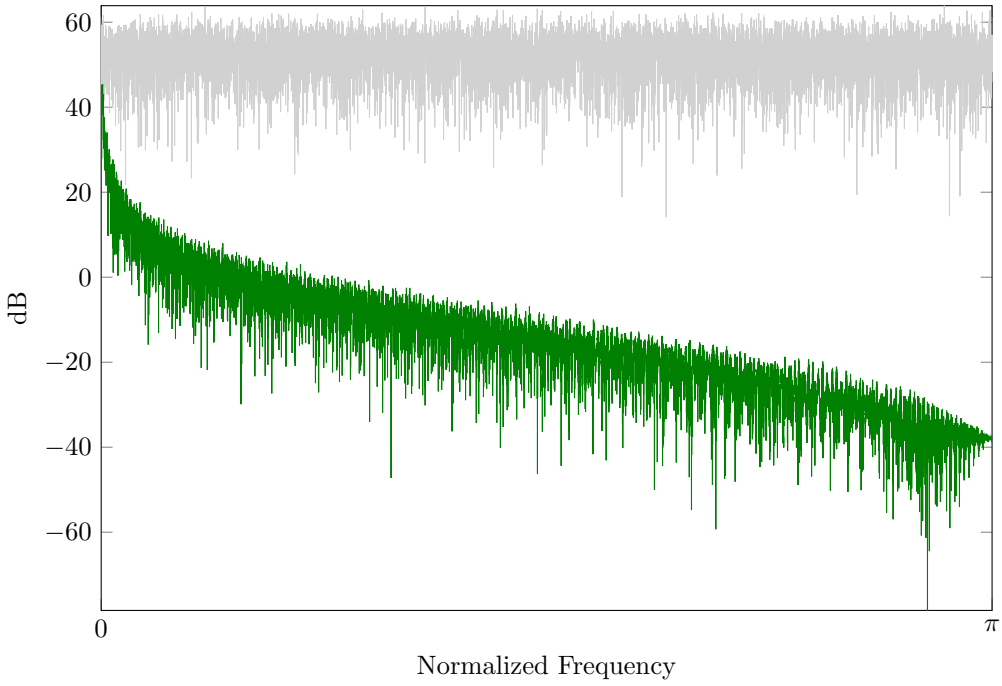


FIGURE 49. Power spectral density of sporadic noise and residual error

(using variance auto-weighting scheme), shown in Figure 48. The reduction in error magnitude is immediately evident; less evident is the relative reduction of noise at different frequencies. Figure 49 provides a power spectral density plot to answer that question. With 40 dB or better rejection at all except the lowest frequencies, the noise reduction rivals that of a high-quality lowpass filter – and VIVA has not removed high frequency components from the true signal.

Successful application of variance auto-weighting to signals corrupted by sporadic noise also indicates that the auto-weighting method does not require that the window size for variance calculation match the burst duration.

4.3. Case Study, Load Cell Monitoring Resuscitation of Hemorrhagic Shock

The four channel "smart IV pole" system (Sparx Engineering, Manvel, TX) previously described has been used for monitoring fluid balance during animal studies at the UTMB Resuscitation Research Laboratory. Many experiments study responses of the circulatory system in injury-induced shock states. Flow rates were simultaneously measured using an electronic Doppler transit-time flowmeter (Transonic Systems) for comparison and confirmation of results.

Figure 50 shows the IV bag weight data logged by the load cells of the smart IV pole during one such study which induced hypovolemia shock in a 40 kg swine via controlled hemorrhage, then resuscitated the subject alternately using Lactated Ringer's solution and blood. One IV pole channel was not used during this study. In the other three channels one can see the increasing weight of the blood storage bags attached to channels 2 and 4 during the controlled hemorrhage near T = 45 min. Later one can see the weight smoothly decreasing as the blood is reinfused, or when Lactated Ringer's is infused. Substantial noise is also visible, introduced by such causes as

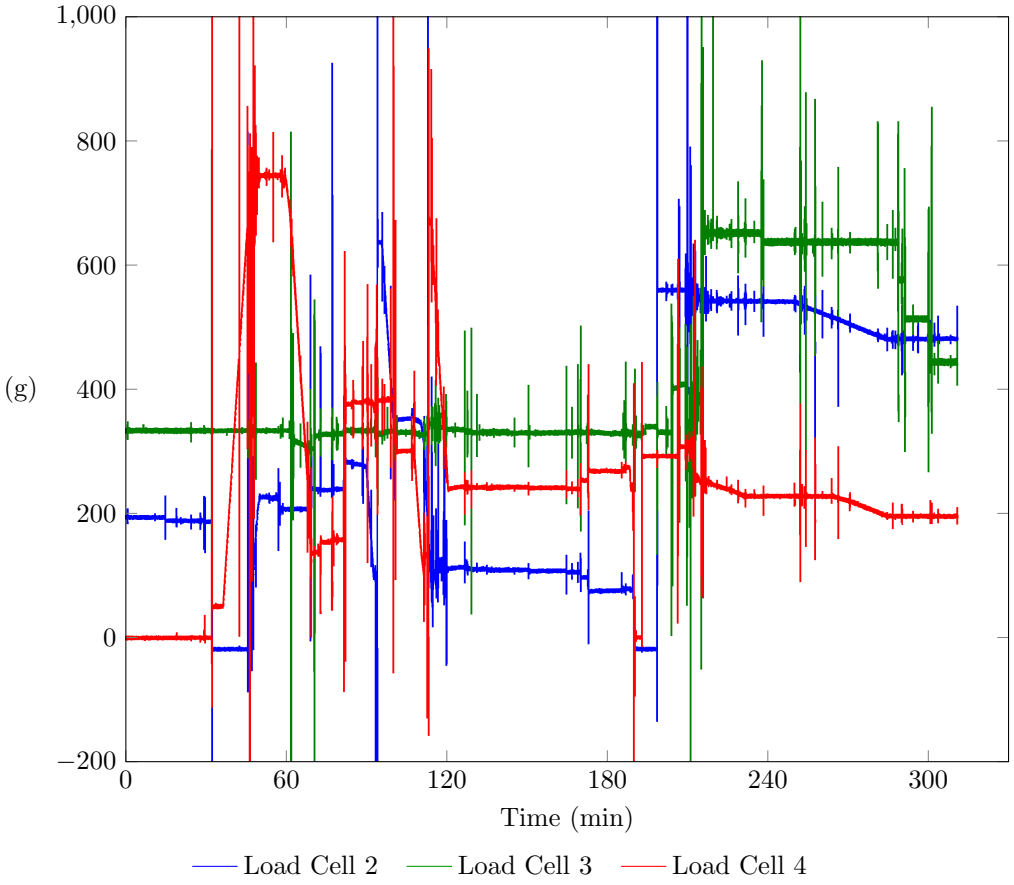


FIGURE 50. Load cell data monitoring pig hemorrhage and resuscitation

replacing depleted bags, moving bags between channels, tension on IV tubing during downstream manipulation, and accidental physical contact with the bags or pole.

Figure 51 shows the series of connected segments identified by AW-VIVA during denoising. The effects of artifacts have been dramatically reduced, but the result still reflects physical actions which change the force on the load cell, such as adding a fresh IV bag, or attaching IV tubing, as well as actual flows. Several heuristics are then applied to reject unreasonable or insignificant changes in weight, resulting in the flow rates of Figure 52.

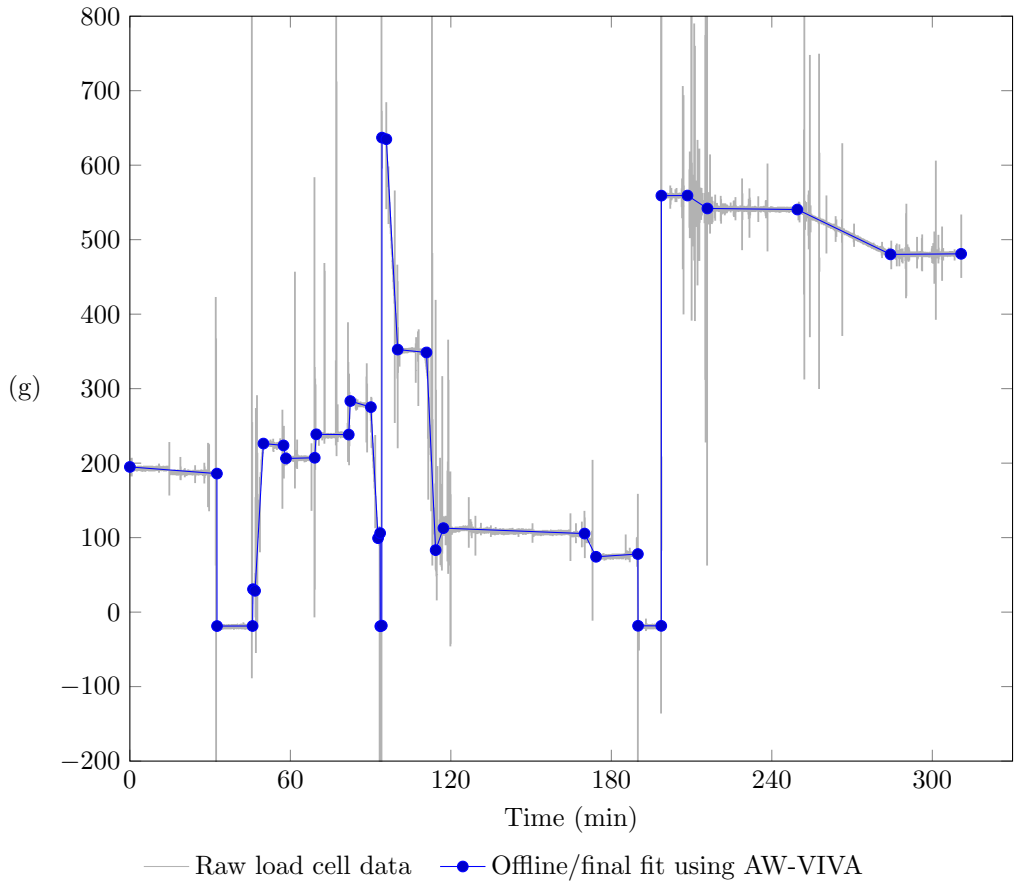


FIGURE 51. Piecewise linear estimate obtained to denoise load cell channel 2, Figure 50

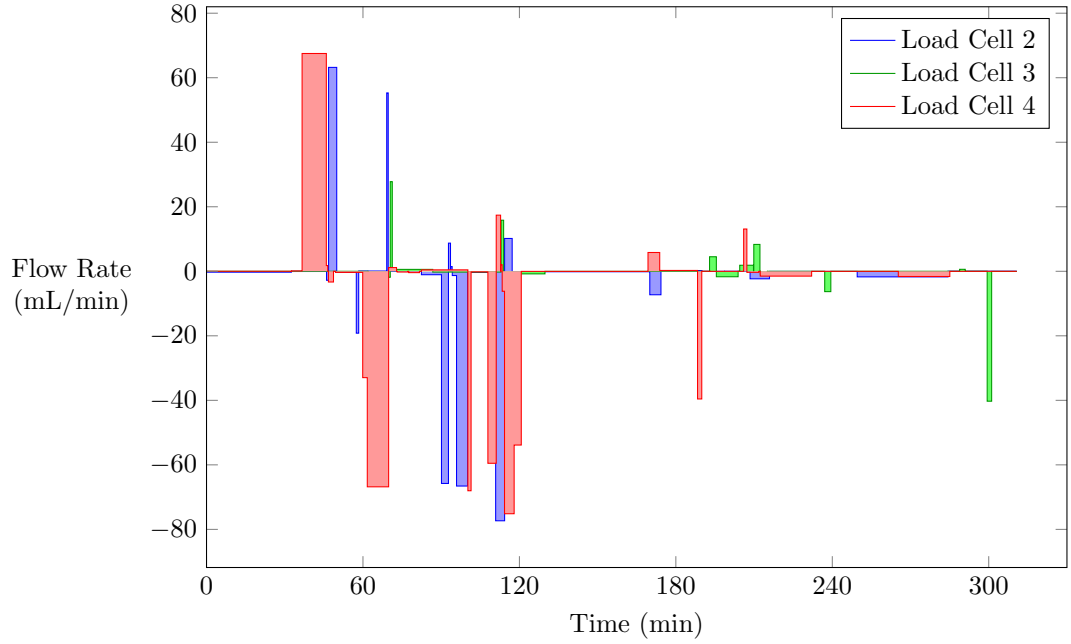


FIGURE 52. Flow rates obtained from piecewise linear estimate

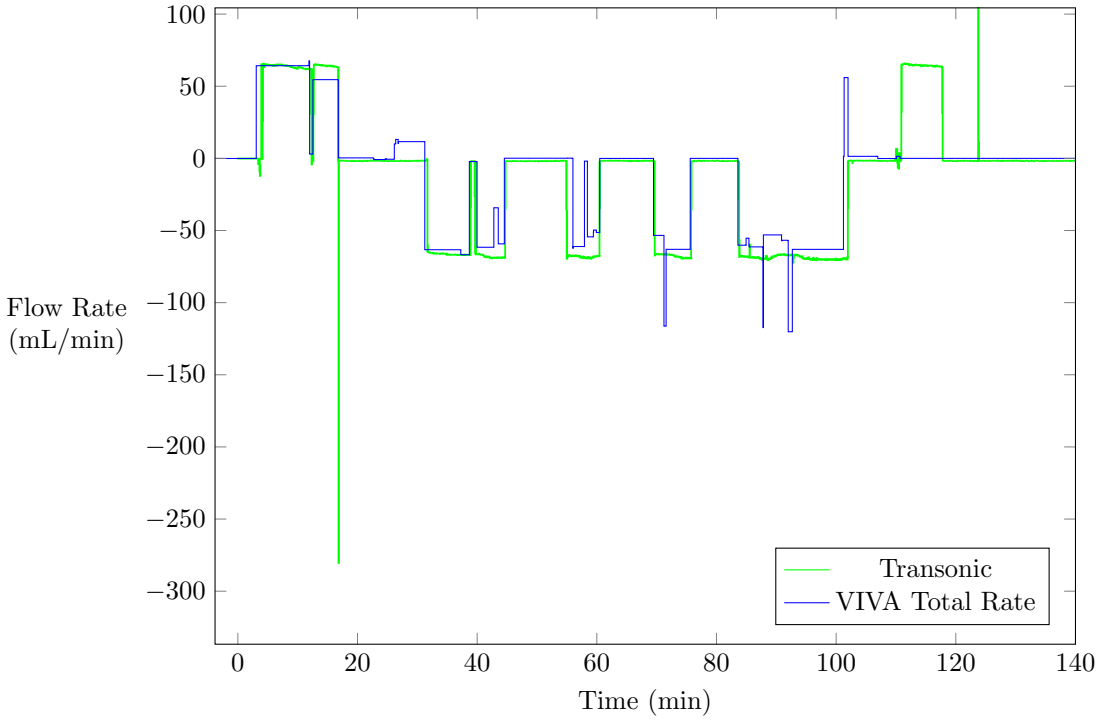


FIGURE 53. Total flow rate from multiple load cells, comparison to electronic Doppler flowmeter

CHAPTER 5

Error and Delay in Online Configurations

5.1. Zero-Delay Evaluation

5.1.1. Uniform Weighting Does Not Inherently Require Delay. Because the VIVA estimator produces a vector of parameter values for the current segment, the piecewise curve can be evaluated at any point in the past, at the present instant, or even used to predict the future (under the assumption of no intervening changepoints). The latter might be applicable for closed-loop feedback control in the presence of an unknown but time-invariant actuator delay: changepoint identification could be used for estimation of the delay, forecasts would be valid because the system would know whether a control command has been recently sent, and the future projection used to make control decisions. (In such a scenario, branching could be confined to occur only during periods following control actions.)

5.1.2. Zero-Delay misses transitions. Transition regions are difficult for zero-delay estimators as seen in Figures 54 and 55. Without considering a sufficient amount of signal following the change, there is no evidence to justify paying the penalty ζ, so the leading path does not reflect presence of the changepoint. For this reason delay is artificially introduced, so that the effect of a number of subsequent samples is included in the path cost. While increasing the number

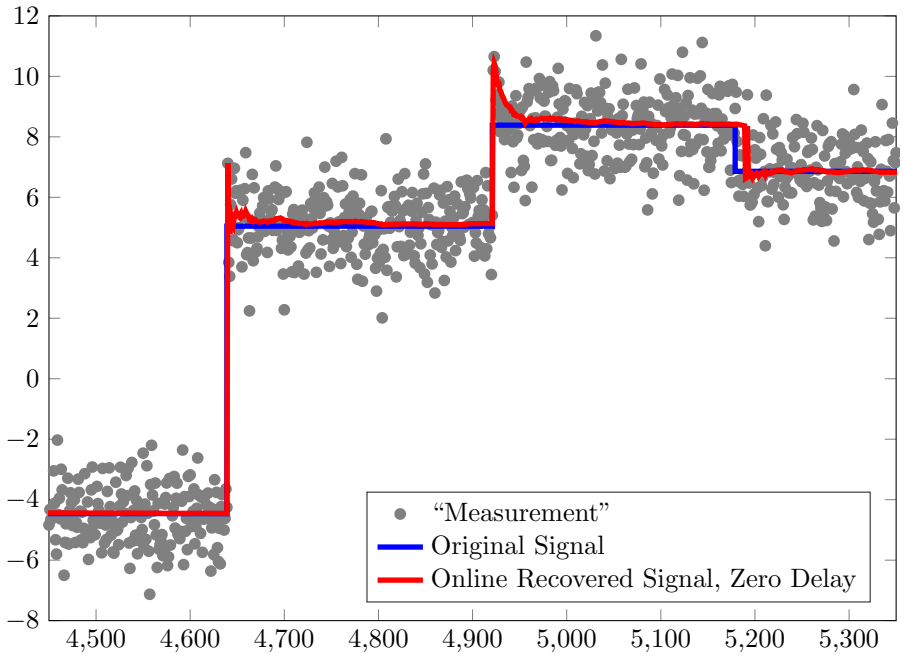


Figure 54. Excerpt from '`piecewise_constant.mat`' and corresponding zero-delay estimate (Uniformly weighted, $\zeta = 25$)

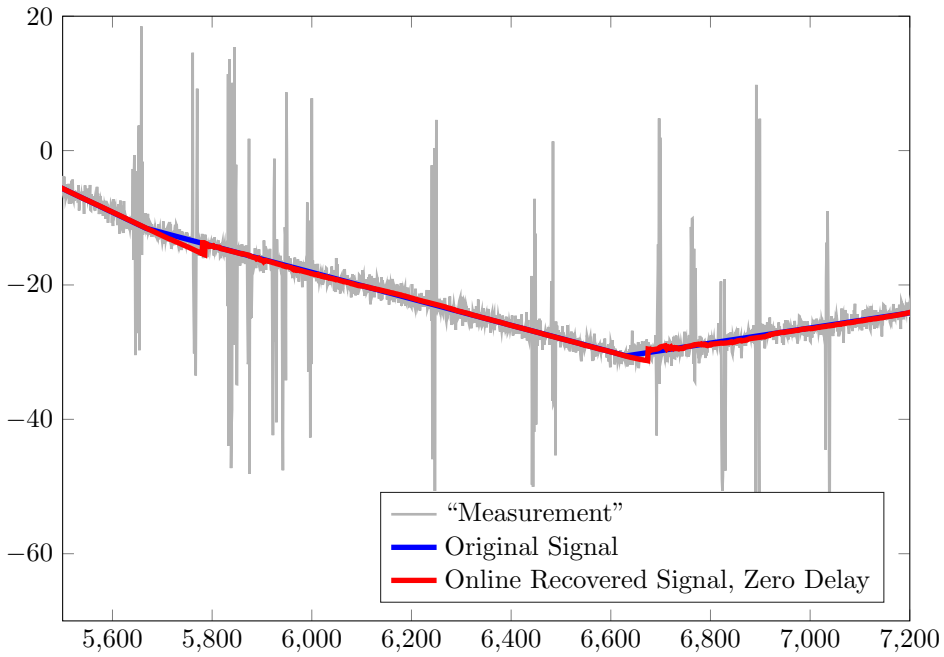


FIGURE 55. Excerpt from '`piecewise_linear_bursty.mat`' and corresponding zero-delay estimate (Variance auto-weighted, $\zeta = 25$)

of "future" samples available for estimation is helpful everywhere, the most profound effect is observed around transitions, where those future samples form the basis for correctly inferring a transition.

5.2. Changepoint Detection Lag

Two approaches are taken to investigation of detection lag: algebraic and empirical. The empirical method is preferred when variance auto-weighting is used, because the weights are difficult to predict.

5.2.1. Solution Stability Condition

Consider again the piecewise constant bicriterion model

$$\begin{array}{lll} \text{minimize} & \sum_i (x'_i - y_i)^2 + \zeta \sum_i b_i & \\ \text{subject to} & \left|x'_{i+1} - x'_i\right| \leq M b'_i & \forall i \\ & b'_i \in \{0, 1\} & \forall i \end{array} \tag{5.1}$$

Assume that $\mathbf{b}' = \mathbf{b}$, that is, the binary variables are chosen according to 1.6 using the true control timing.

Then considering any control time $\mathcal{T}_k$, then the optimization will eliminate that input step unless (compare Figures 56 and 57)

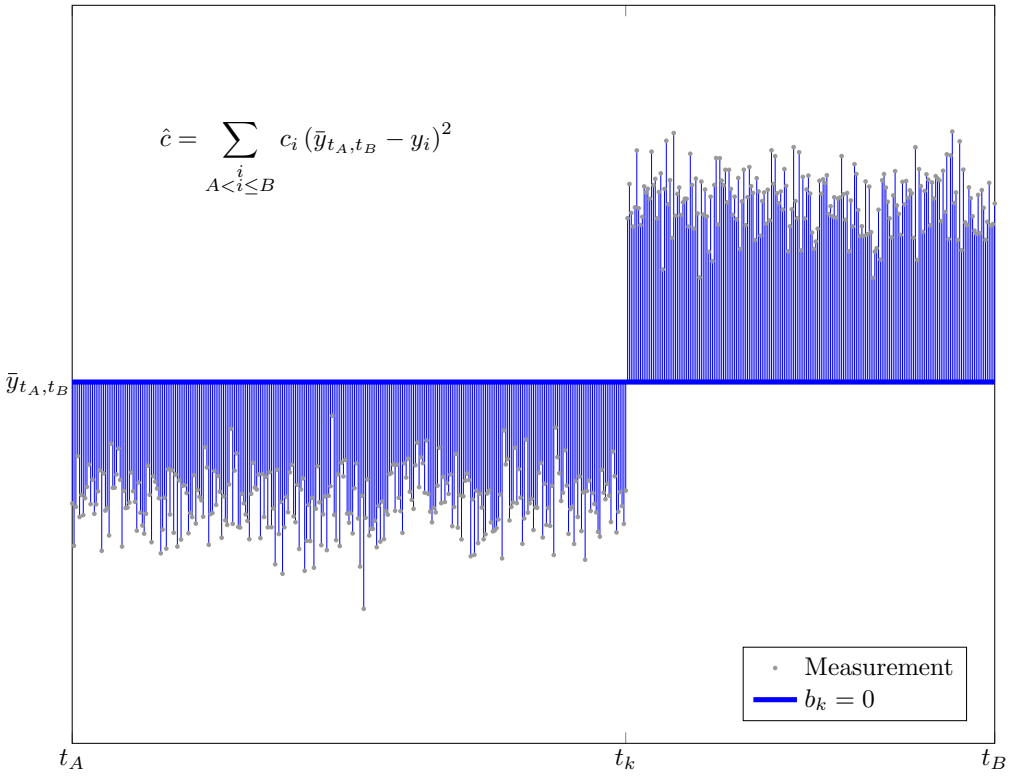


FIGURE 56. Cost function within $b_k = 0$ branch

$$\sum_{\mathcal{T}_{k-1}<t_i\leq\mathcal{T}_{k+1}} c_i \left(\bar{y}_{\mathcal{T}_{k-1},\mathcal{T}_{k+1}} - y_i\right)^2 \geq \sum_{\mathcal{T}_{k-1}<t_i\leq\mathcal{T}_k} c_i \left(\bar{y}_{\mathcal{T}_{k-1},\mathcal{T}_k} - y_i\right)^2 + \zeta + \sum_{\mathcal{T}_k<t_i\leq\mathcal{T}_{k+1}} c_i \left(\bar{y}_{\mathcal{T}_k,\mathcal{T}_{k+1}} - y_i\right)^2 \tag{5.2a}$$

where $\bar{y}_{t_A,t_B}$ is the (weighted) block temporal mean

$$\bar{y}_{t_A,t_B} = \Big(\sum_{t_A<t_i\leq t_B} c_i\Big)^{-1} \sum_{t_A<t_i\leq t_B} c_i y_i \tag{5.2b}$$

But (this is also the formula relating moment of inertia of a planar mass distribution about its centroid to moment of inertia about another parallel axis)

$$\sum_{t_A<t_i\leq t_B} (\eta - y(\tau))^2 = (t_B - t_A)\left(\eta - \bar{y}_{t_A,t_B}\right)^2 + \sum_{t_A<t_i\leq t_B} \left(\bar{y}_{t_A,t_B} - y_i\right)^2 \tag{5.2c}$$

so that in the uniformly weighted case Equation 5.2a becomes

$$\zeta \leq (\mathcal{T}_k - \mathcal{T}_{k-1})\left(\bar{y}_{\mathcal{T}_{k-1},\mathcal{T}_{k+1}} - \bar{y}_{\mathcal{T}_{k-1},\mathcal{T}_k}\right)^2 + (\mathcal{T}_{k+1} - \mathcal{T}_k)\left(\bar{y}_{\mathcal{T}_{k-1},\mathcal{T}_{k+1}} - \bar{y}_{\mathcal{T}_k,\mathcal{T}_{k+1}}\right)^2 \tag{5.2d}$$

$$\zeta \leq \frac{(\mathcal{T}_k - \mathcal{T}_{k-1})(\mathcal{T}_{k+1} - \mathcal{T}_k)}{\mathcal{T}_{k+1} - \mathcal{T}_{k-1}} \left(\bar{y}_{\mathcal{T}_{k-1},\mathcal{T}_k} - \bar{y}_{\mathcal{T}_k,\mathcal{T}_{k+1}}\right)^2 \tag{5.2e}$$

The fraction is bounded by

$$\frac{1}{2}\min(\mathcal{T}_k - \mathcal{T}_{k-1}, \mathcal{T}_{k+1} - \mathcal{T}_k) \leq \frac{(\mathcal{T}_k - \mathcal{T}_{k-1})(\mathcal{T}_{k+1} - \mathcal{T}_k)}{\mathcal{T}_{k+1} - \mathcal{T}_{k-1}} \leq \min(\mathcal{T}_k - \mathcal{T}_{k-1}, \mathcal{T}_{k+1} - \mathcal{T}_k)$$

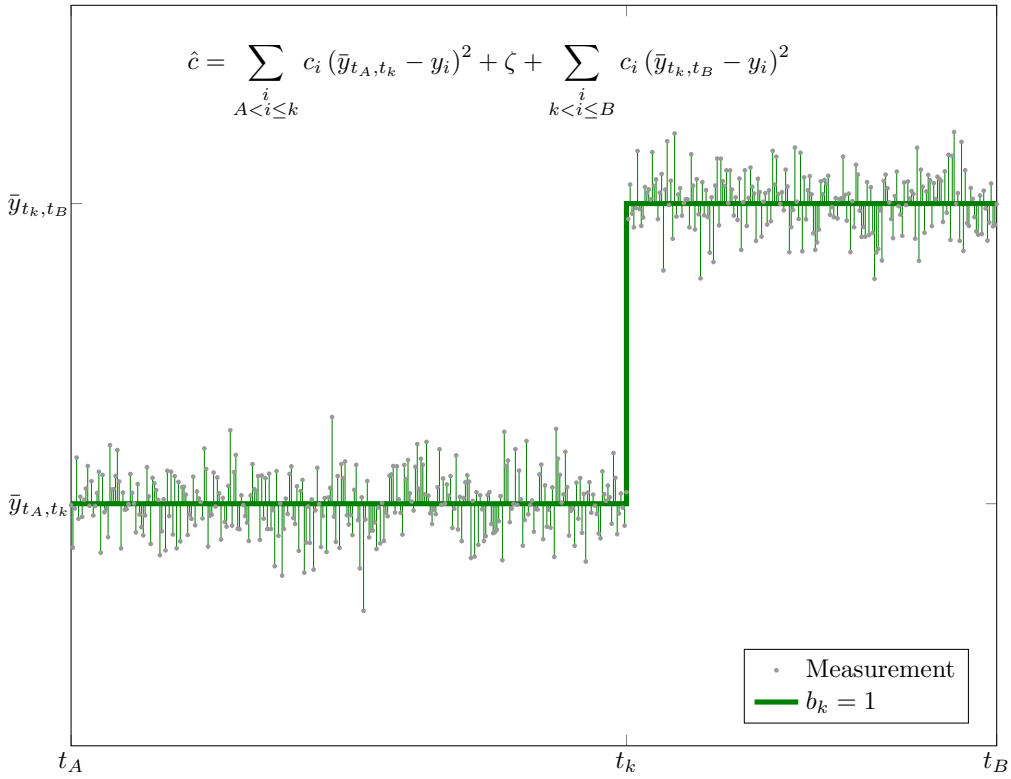


FIGURE 57. Cost function within $b_k = 1$ branch

which suggests that a good choice for ζ is

$$\zeta_{ideal} = \frac{1}{2} t_{th} y_{th}^2 \tag{5.3}$$

where t_{th} and y_{th} are the desired detection thresholds for time and magnitude, respectively.

5.2.2. Selecting Delay Based on Solution Stability

When designing for online operation, the *effective* segment duration is limited to the delay time, and Equation 5.3 should be updated correspondingly:

$$\zeta_{ideal} = \frac{1}{2} t_{delay} y_{detect}^2 \tag{5.4}$$

Naturally, if ζ has already been selected, this formula may be solved for the required t_{delay} to achieve a desired detection sensitivity y_{detect}, and the equation can also be used to predict the sensitivity of a system with fixed ζ and t_{delay}.

Changes which are more subtle than y_{detect} may be expected to converge late, and oscillation between paths with and without a detected changepoint may occur. Specifically, noise amplification may be observed in the interval

$$t - \mathcal{T}_k < t_{detect} - t_{delay} \tag{5.5a}$$

where

$$t_{detect} = 2\frac{\zeta}{(y_k - y_{k-1})^2} \tag{5.5b}$$

5.2.3. Selecting Delay Based on Observed Population Reduction

For models with sloped or curved segments, the analytic expressions become more difficult to manipulate. When confidence-based weighting is added to the mix, finding an analytic expression may be infeasible due to unknown noise characteristics. In such cases an empirical approach to delay selection is desired.

One such empirical approach follows from the insight that the reason for incorrect estimates following changepoints, and particularly for oscillation, is due to the existence of paths both with and without detected changepoint in the population of candidate paths. By inspecting the relationship between population size and time elapsed since change, the time at which paths which fail to acknowledge the change are removed becomes apparent. From this an appropriate time delay for online processing may be inferred.

5.3. Online Denoising Performance

5.3.1. Piecewise-constant with additive i.i.d. Gaussian noise

Online denoising of is performed using a variety of filters and the VIVA estimator:

- A 60-sample equally weighted moving average, $H_{ma}(z) = \frac{1}{60}\sum_{i=0}^{59} z^{-i}$
- A single pole IIR filter, $(1 - .97z^{-1})H_{sp}(z) = .03$
- A 511-tap sinc FIR lowpass filter, $|H_{lp}(f/F_s)| = u(.005 - |f/F_s|)$
- A median filter using a window of $N = 51$ samples
- Estimate of the current sample, using the lowest cost (to-date) path found by VIVA using uniform weighting and $\zeta = 25$
- Estimate of 15 samples prior, using the lowest cost (to-date) path found by VIVA

In addition, a second regularization formulation using non-uniform weighting is compared, also optimized using VIVA. The weights are automatically determined from the input signal, and chosen to be inversely proportional to the local variance (10 samples before and after the current position) in the noisy measurement signal. This introduces an additional 10 sample latency.

All these signals can be computed in near-realtime, with a system delay varying from 0 lag for the single pole IIR and current sample uniformly-weighted VIVA, to 25 sample delay for the variance-weighted 15-step-behind VIVA and median filter, to 255 sample delay for the sinc lowpass filter.

The resulting signal using each method is plotted in Figure 58.

Clearly the estimate given by VIVA is better in terms of both steady-state residual error and behavior near step changes, as long as the assumption of large sparse variations in the original signal is valid. In fact, the uniformly-weighted VIVA estimator with 15 samples of delay is visually indistinguishable from the original signal.

In addition to visual verification on a case-by-case basis that the results are good, quantitative performance metrics on a set of 240 randomly synthesized signals are shown in Figures 59 and 60.

Also note that a sharper frequency cutoff for the lowpass filters does not lead to better performance, because the step changes in the original signal are sharp and have wideband frequency content.

5.3.2. Piecewise-constant with burst Gaussian noise

Figures 62 and 63 show outcomes from the same online estimators as were used with i.i.d. noise. The lowpass filters turn out nearly the same results as to maximum absolute error, which is still dominated by smoothing steps in the input signal. However, the filters are letting some of the increased noise through in the stable regions, which is hardly surprising.

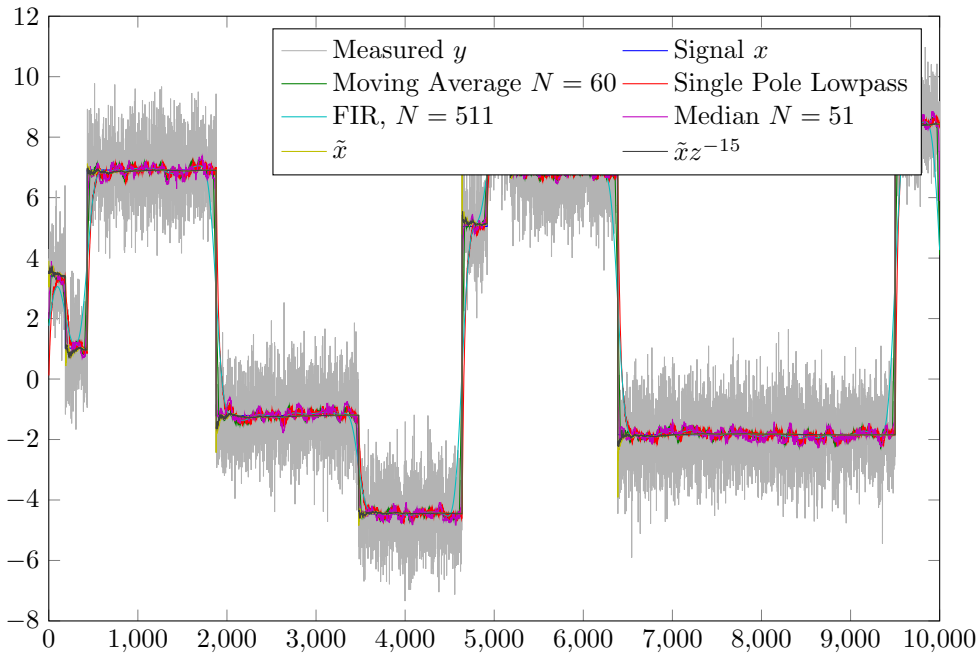


FIGURE 58. Denoising piecewise_constant.mat using conventional lowpass filters and VIVA

Just as with post-processing in the presence of burst noise, online runs of the variance-weighted VIVA method, which in the i.i.d. noise case yielded estimates slightly inferior to its uniform weighting sibling, now show a huge advantage, delivering an order of magnitude lower residual error than all competitors, excepting only the maximum absolute error. Its ability to assign low weights to transient changes have given it effective immunity to the burst noise.

5.3.3. Piecewise-linear with i.i.d. Gaussian noise

Just as for piecewise constant signals, delaying polyline estimation until VIVA has analyzed a number of future samples, made possible by introducing delay, yields the greatest benefit at and immediately after transitions. And consistent with retrospective analysis, online denoising of piecewise linear signals misses fewer steps due to variance weighting than did piecewise constant estimators. VIVA running online is still subject to large maximum errors in the rate signal (Figure 66), just as the offline analysis was. The conventional filtering mechanisms do not particularly benefit from low variance near steps, however. Rejection of high-frequency noise is opposed by the frequency-proportional response of the derivative, and residual errors exist throughout (Figure 67), not merely near transitions.

The error bands in both value and rate estimates are also considerably wider than in the piecewise-constant case, because the randomly chosen signals exhibit a wide range of slopes.

5.3.4. Piecewise-linear with burst white Gaussian noise

Figures 68 and 70 show that the auto-weighting VIVA estimates with 15 and 60 sample delays lead in every error metric, except for maximum rate error where the sharp-cutoff FIR filters match their performance. This outcome is really not very surprising, considering that the various features of VIVA were designed for this type of input.

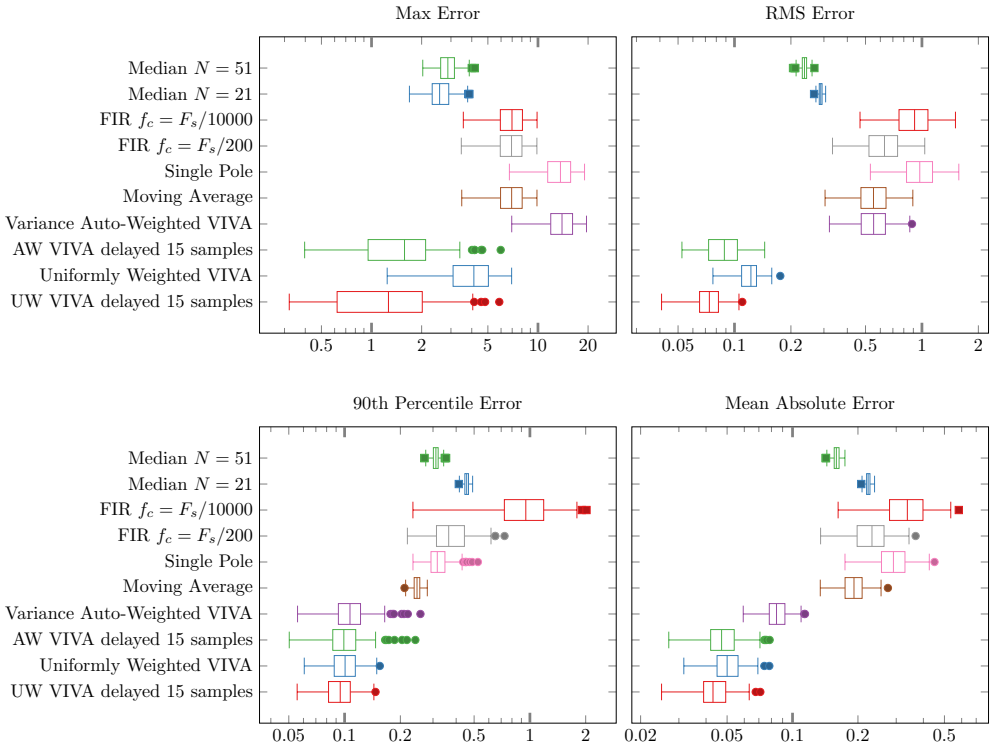


FIGURE 59. Performance of online estimators evaluated using piecewise constant signal (strongly white noise), N = 240

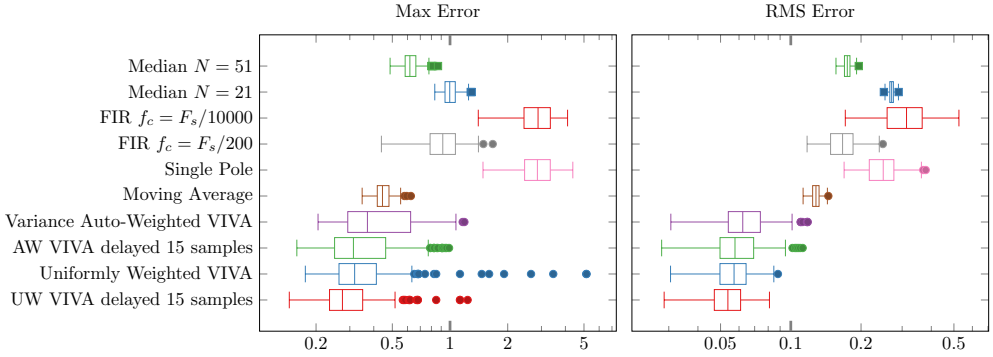


FIGURE 60. Performance in stable regions (transition bands excluded) of online estimators evaluated using piecewise constant signal (strongly white noise), N = 240

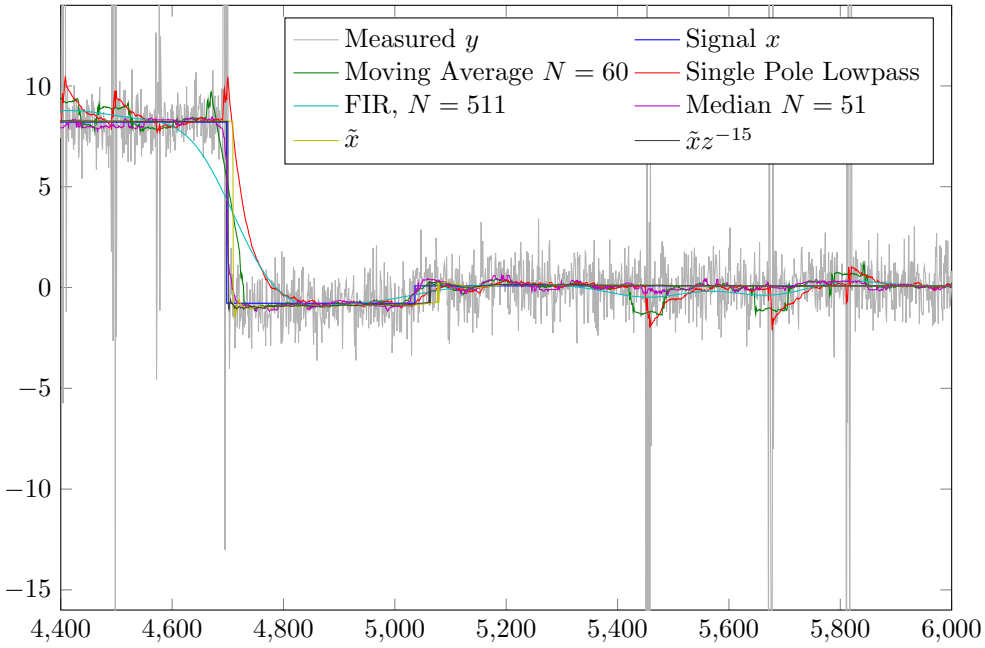


FIGURE 61. Denoising piecewise_constant_bursty.mat using conventional low-pass filters and Variance Auto-Weighted VIVA

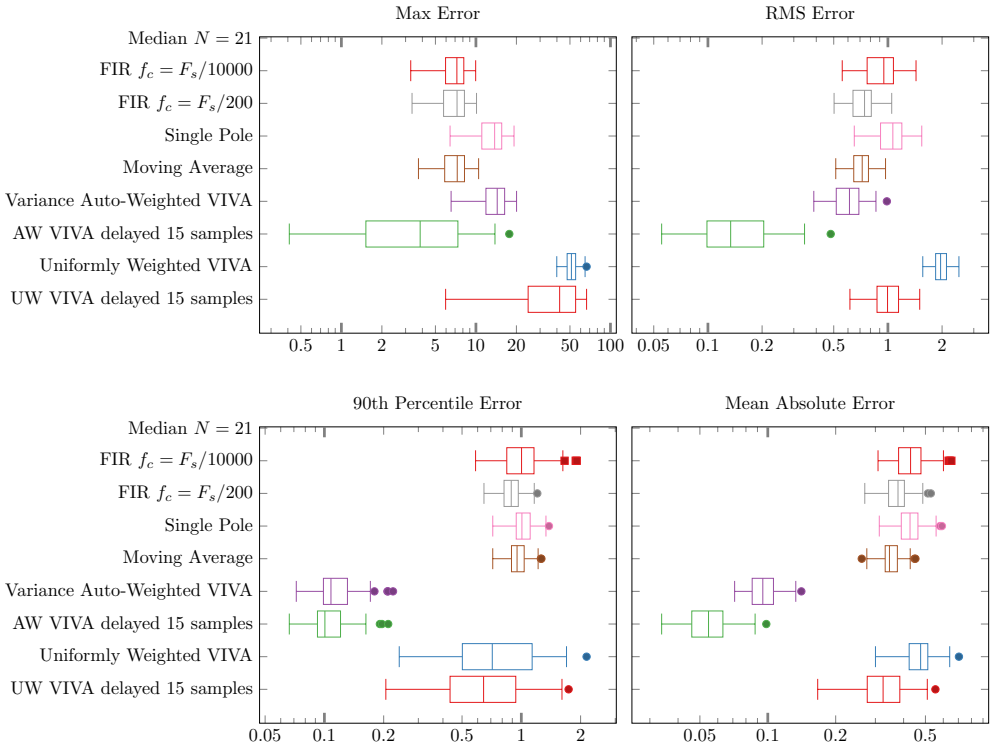


FIGURE 62. Performance of online estimators evaluated using piecewise constant signal (burst noise), N = 100

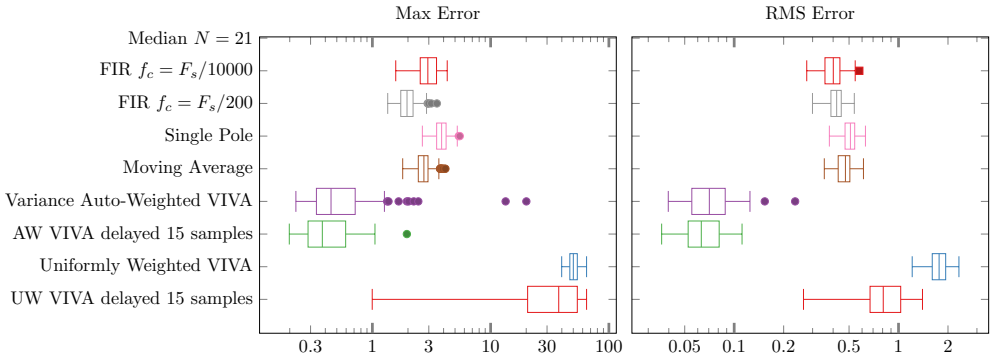


FIGURE 63. Performance in stable regions (transition bands excluded) of online estimators evaluated using piecewise constant signal (burst noise), N = 100

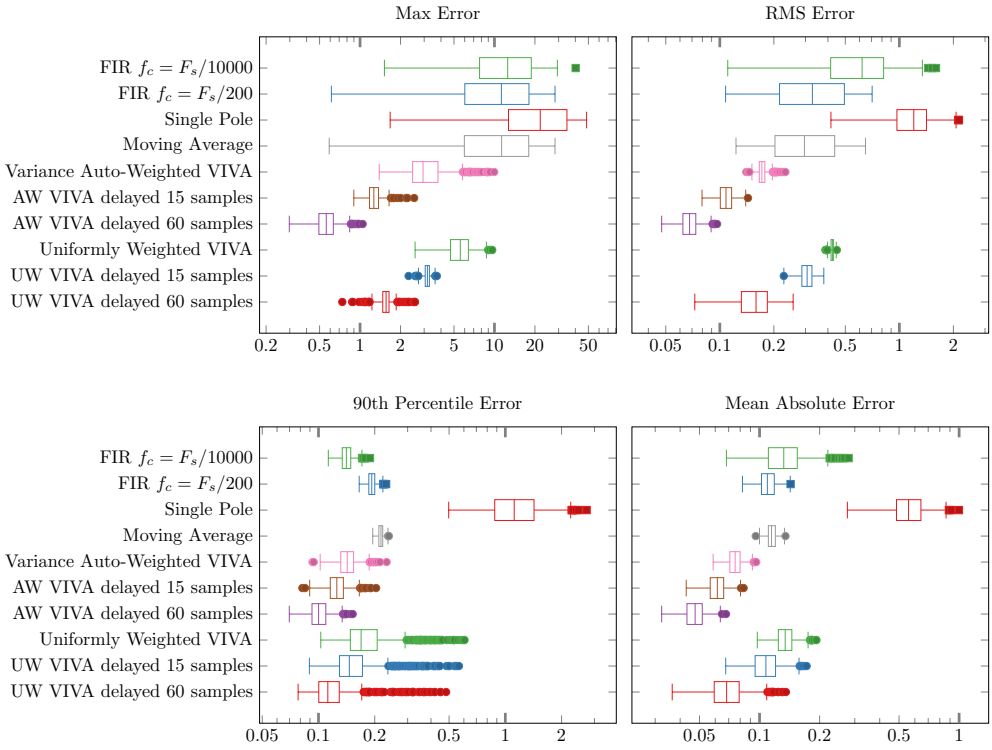


FIGURE 64. Performance of online estimators evaluated using piecewise linear signal (i.i.d. white noise), N = 500, analysis of value estimate

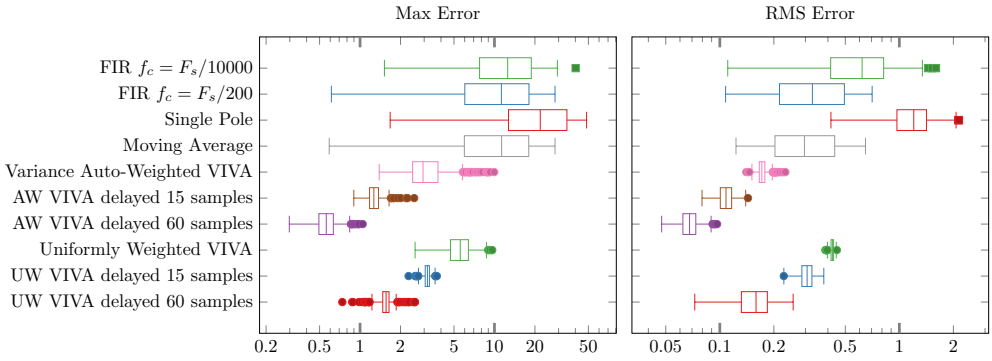


FIGURE 65. Performance in stable regions (transition bands excluded) of online estimators evaluated using piecewise linear signal (i.i.d. white noise), N = 500, analysis of value estimate

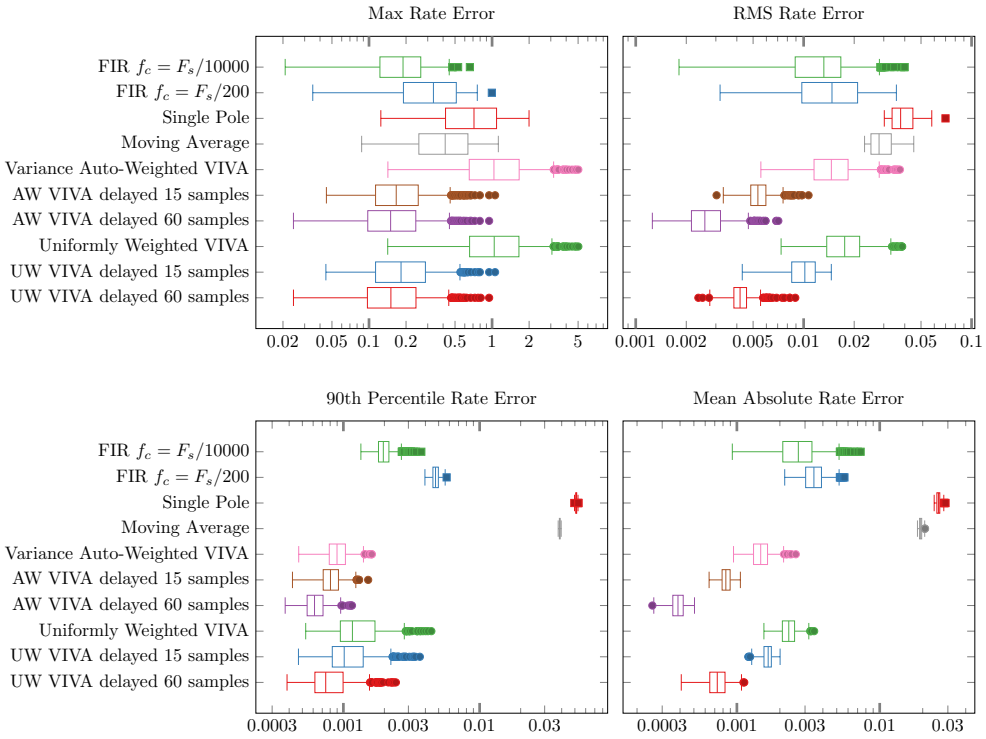


FIGURE 66. Performance of online estimators evaluated using piecewise linear signal (i.i.d. white noise), N = 500, analysis of rate estimate

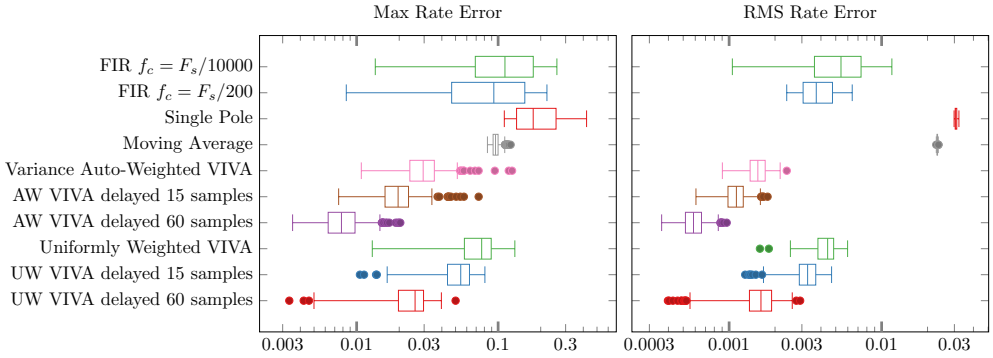


FIGURE 67. Performance in stable regions (transition bands excluded) of online estimators evaluated using piecewise linear signal (i.i.d. white noise), N = 500, analysis of rate estimate

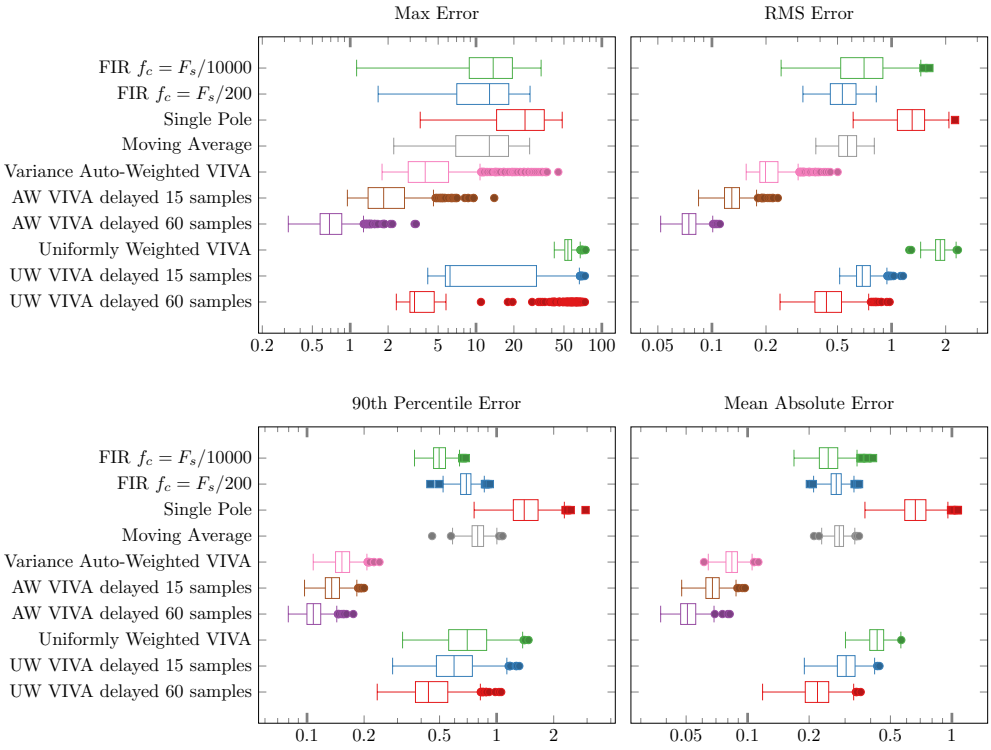


FIGURE 68. Performance of online estimators evaluated using piecewise linear signal (burst noise), N = 500, analysis of value estimate

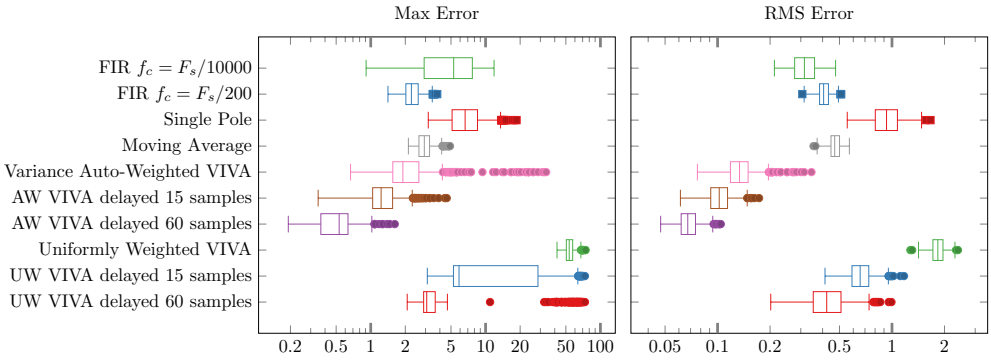


FIGURE 69. Performance in stable regions (transition bands excluded) of online estimators evaluated using piecewise linear signal (burst noise), N = 500, analysis of value estimate

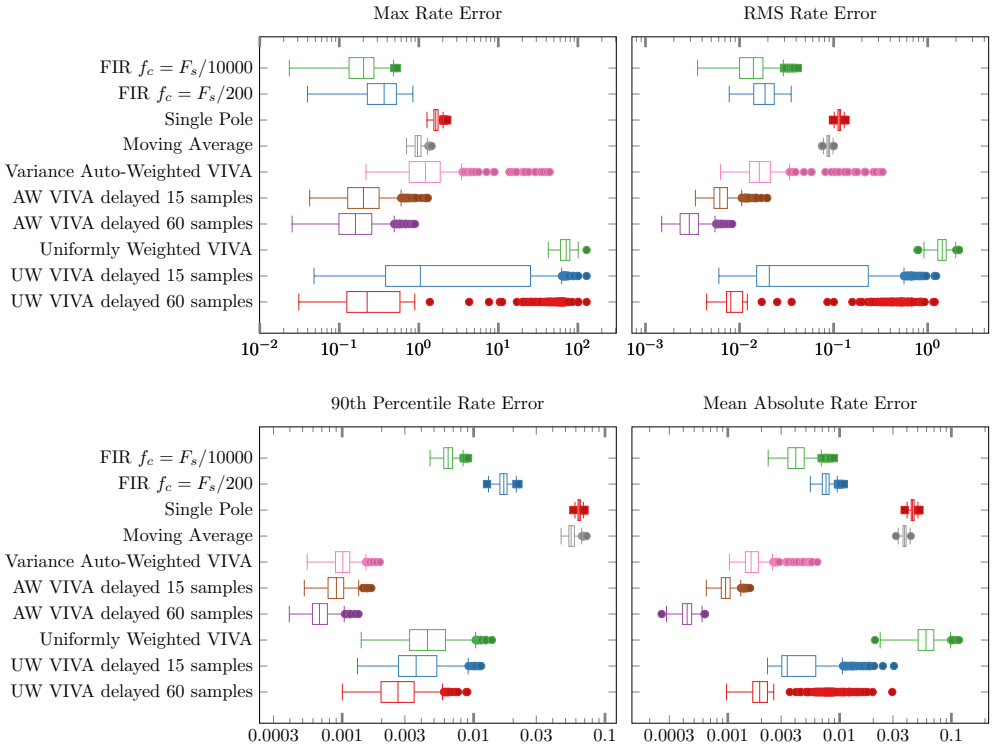


FIGURE 70. Performance of online estimators evaluated using piecewise linear signal (burst noise), N = 500, analysis of rate estimate

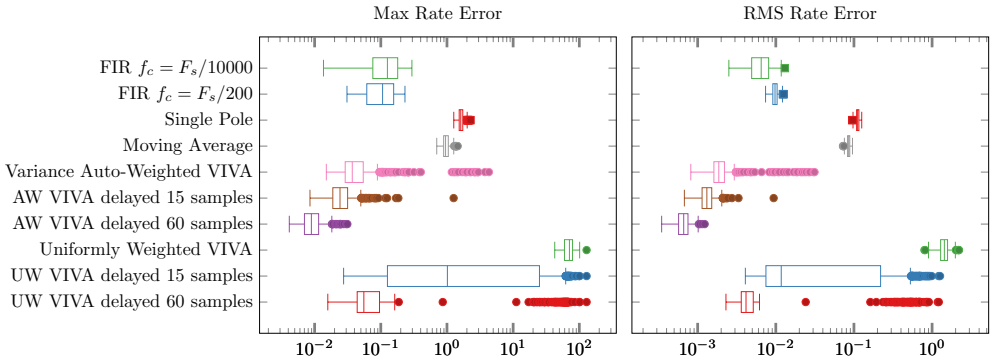


FIGURE 71. Performance in stable regions (transition bands excluded) of online estimators evaluated using piecewise linear signal (burst noise), N = 500, analysis of rate estimate

CHAPTER 6

Conclusions

6.1. Summary

The objective of this dissertation has been to advance the state of critical care automation and decision support systems by reducing measurement error and this improving the fidelity of the control system's situational awareness of patient status and response dynamics. Specifically, a new approach has been demonstrated for minimal cardinality recovery of two broad categories of segmented signals: piecewise-separable, embodied in the piecewise constant model, and piecewise-continuous, embodied by the piecewise linear segments model.

First, these problems were expressed as a mixed-integer program with a quadratic goal function and linear and binary constraints. A naive solution to such a problem must elaborate all combinations of the binary variables in exponential time. The VIVA algorithm draws upon two known techniques for exploring binary sequences: branch-and-bound optimization of mixed-integer programs and Viterbi decoding of convolutional-coded digital messages. These ideas are adapted to the step-fitting and trend-fitting problems using a novel pruning constraint. A guaranteed bound of quadratic time was proven for piecewise-separable problems, but this worst case only occurs when the optimal solution contains a single segment. Processing time for many problems scales linearly.

6.2. Impact

The VIVA algorithm may immediately be used in conjunction with load cells to create a low-cost mechanically robust hospital infusion monitoring system with accuracy that rivals that of dedicated laboratory flowmeters or dedicated research staff and far higher accuracy with respect to volume, flow rate, event timing, and flow duration than manual notes routinely collected by nursing personnel. The resulting high quality records would enable research and quality improvement initiatives to answer questions about protocol compliance, frequency of flow cessation due to bag depletion, and response time to alarms. In addition, the fluid records may be paired with high resolution vital sign recordings for model development and validation work.

Because VIVA directly optimizes for sparse descriptions, it is very useful for lossy compression. Varying the regularization parameter controls the compression level, trading off the detail level of the estimate against the increased size of the representation. Piece-wise descriptions are especially useful for streaming data to mobile devices, not only because the sparse representation consumes less network bandwidth, but device memory requirements are reduced. If a piece-wise linear representation is chosen, the vertices of a polyline can be directly rendered by OpenGL ES hardware-accelerated graphics.

VIVA may also be used as a general-purpose step fitting algorithm, applicable to problems such as the study of protein assembly dynamics which motivated the precursor algorithms of Kerssemakers et al. (2006); Little and Jones (2010). Although VIVA has higher computational requirements than competing offline methods, it performs analysis online in near real-time at no

additional cost. Even when online analysis is not required, the additional calculation may be justified by the improved fit or better sparsity.

6.3. Whither?

An obvious application of the VIVA algorithm would be the original intended use – providing accurate flow rate measurements to the infusion effects monitor in near real-time. Major remaining steps include parallel analysis of data using flow rate information from both the electronic Doppler flowmeters and the load cells with variance auto-weighted VIVA rate estimation, quantifying the effect of residual input measurement errors on the adaptive filters and parametric fit in blood pressure response models under development, and finding the VIVA parameters (regularization weight ζ and time delay) that minimize these effects.

In addition, it may be possible to improve the VIVA online estimate by considering multiple top paths. For example, the variance of estimates made by the top k paths, as well as agreement between estimates made with different delays, could lead to a measure of confidence in VIVA's estimate output, which in turn could be used for weighting fitting error in the downstream model.

Analysis of sensitivity to the regularization parameter can make use of the solution stability criterion, if one can be found for the piecewise-linear case. Better pruning criteria for piecewise linear signals could replace the current set of path discard heuristics and improve the runtime-vs-optimality tradeoff. Analyses could include the frequency and magnitude of error in transition timing recovery as an additional performance metric. Also, while the additive white Gaussian noise models are convenient for analysis and often justified by the Central Limit Theorem, in the particular application to IV bags hanging from load cells, underdamped pendulum action leads to a natural frequency of oscillation (which changes slowly as the bag empties), and following a physical impact, rather than abruptly ceasing as in the burst noise model, the interfering factors undergo exponential decay. VIVA performance under these noise conditions could be studied by characterizing the underdamped oscillation and augmenting the white noise model with pendulum interference.

Finally, optimization of the implementation code could reduce constant factors in running cost, both time and space. Even though asymptotic complexity is not improved, such tuning may make the algorithms suitable for embedding into onboard battery-powered electronics of sensors such as load cells, instead of requiring the resources of a mainstream desktop computer system.

In addition to use as independent confirmation and calibration of fluid infusion rate delivered from hospital IV pumps, where a piecewise-linear description fits the data well, there are exciting possibilities for use of VIVA's underlying concepts with data fitting other descriptions. Polynomial segments may be straightforwardly fit as described in Kim et al. (2009) by making the obvious substitution of the ℓ^0 norm in place of ℓ^1. Perhaps the conjoined segments individually are not affine but described by decaying exponentials, as is the case with infusion rates delivered via gravity flow or pressure bag, since in both cases the driving pressure decays as the bag empties.

A very powerful technique could be developed through combining VIVA and adaptive filtering to address unknown changes in system dynamics, by mapping the recovered binary sequence onto model selection. For example, in a Kalman filter, whether process noise has low or high variance can be expressed as a binary variable. In orthogonal least-squares filtering, the binary variable can control covariance resetting. More generally, the binary variable shifts the estimator between two modes: target-seeking and target-tracking, and VIVA provides a mechanism for "regaining target lock" by repeating the seeking action when the output error justifies it, while using a penalty weight to remain in tracking mode as long as tracking continues giving good results.

APPENDIX A

Selected Matlab and C# Functions

A limited number of functions which may aid in understanding of the text are included here. For electronic distributions of source code and complete input datasets, by means of which the interested reader may reproduce all results, please refer to the companion website:

`http://www.sparxeng.com/viva/`

A.1. Synthesis of test data for Monte Carlo analysis

Monte Carlo analysis is used for characterization of denoising performance throughout, beginning in section 3.6.1. Datasets of "source" signals: step sequences (piecewise constant) and polylines (piecewise linear) were synthesized using the below scripts. For each source signal, the randomly chosen generator parameters were saved. Then each signal was uniformly sampled and Gaussian noise was generated and combined additively to form a "measured" signal.

A dataset was formed for each combination of source signal model (step or polyline) and noise model ("uniform" strongly white or "burst" weakly white), as follows:

Table A.1. Size of Monte Carlo datasets

Source Model	Segment Count	Sample Count	Number of Signals	
			(Uniform Noise)	(Burst Noise)
Step	10	10 000	240	100
Polyline	10	20 000	500	500

Where multiple analyses were performed using the same signal model (for example, online, delay, and retrospective), the same dataset was consistently used, allowing direct comparison of residual noise results.

A.1.1. gen_piecewise_constant.m

Listing A.1 produces test vectors which are total length `N` having `k` piecewise constant segments, each segment having minimum duration 100 samples.

All parameters are optional, if omitted or an empty matrix `[]` is supplied, the default values will be used as follows:

- Default `k` = 10 segments
- Default `N` = 10000 total samples

To achieve the minimum duration of 100 samples per segment, it is necessary that $\texttt{N} \geq 100\texttt{k}$.

LISTING A.1. Generation of Piecewise Constant Signal

```
function [y, mu, changet] = gen_piecewise_constant(k, N)
    if nargin < 2
        N = 10000;
    end
    if nargin < 1
        k = 10;
    end

    changet = (N - k*100) * rand(k-1,1);
    changet = (1:k-1)'*100 + floor(sort(changet));
    changet = [0; changet; N];
    mu = -10 + 20 * rand(k,1);
    y = zeros(N,1);
    for i = 1:k
        [y(changet(i)+1:changet(i+1))] = mu(i);
    end
end
```

A.1.2. gen_piecewise_linear.m

Listing A.2 produces a test vector with `k` connected linear segments, each having duration not less than 100 samples, and a total length of `N` samples.

All parameters are optional, if omitted or an empty matrix `[]` is supplied, the default values will be used as follows:

- Default `k` = 10 segments
- Default `N` = 10000 total samples

To achieve the minimum duration of 100 samples per segment, it is necessary that `N` $\geq$ 100`k`.

LISTING A.2. Generation of Piecewise Linear Signal

```
function [y, b, changet] = gen_piecewise_linear(k, N)
    if nargin < 2
        N = 10000;
    end
    if nargin < 1 || isempty(k)
        k = 10;
    end

    changet = (N - k*100) * rand(k-1,1);
    changet = (1:k-1)'*100 + floor(sort(changet));
    changet = [0; changet; N];
    b = -50 + 100 * rand(k+1,1);
    y = interp1q(changet, b, [.5:N]');
end
```

A.1.3. gen_iid_white_noise.m

Listing A.3 generates a random vector of independent identically distributed (i.i.d.) white Gaussian noise.

The noise signal `w` will be generated with a length of `N` samples. The noise samples share a Gaussian distribution with zero mean and standard deviation `sigma`.

All parameters are optional, if omitted or an empty matrix `[]` is supplied, the default values will be used as follows:

- Default `N` = 10000
- Default `sigma` = 1

LISTING A.3. Generation of i.i.d. Gaussian White Noise Signal

```
1  function [w] = gen_iid_white_noise(N, sigma)
15
16     if nargin < 1 || isempty(N)
17         N = 10000;
18     end
19     if nargin < 2
20         sigma = 1;
21     end
22
23     w = sigma * randn(N,1);
```

A.1.4. gen_burst_white_noise.m

Listing A.4 generates a random vector of weakly white Gaussian burst noise.

The noise signal `w` will be generated with a length of `N` samples. The noise samples are individually zero mean Gaussian distributed. The standard deviation is controlled by `sigma(1:2)`.

The samples are grouped into bursts of `r` adjacent samples, which share the same standard deviation according to:

$$\sigma = \begin{cases} \texttt{sigma(1)} & \text{with probability } \texttt{p} \\ \texttt{sigma(2)} & \text{with probability } (1 - \texttt{p}) \end{cases} \tag{A.1}$$

All parameters are optional, if omitted or an empty matrix `[]` is supplied, the default values will be used as follows:

- Default `N` = 10000 total samples
- Default `r` = 10 samples per burst
- Default `p` = 0.95 chance of using `sigma(1)`
- Default `sigma` = $\begin{bmatrix} 1 & 16 \end{bmatrix}$

LISTING A.4. Generation of Gaussian Burst Noise Signal (Weakly White)

```
function [w] = gen_burst_white_noise(N, r, p, sigma)
    if nargin < 1 || isempty(N)
        N = 10000;
    end
    if nargin < 2 || isempty(r)
        r = 10;
    end
    if nargin < 3 || isempty(p)
        p = .95;
    end
    if nargin < 4
        sigma = [1 16];
    end

    prbs = repmat(rand(1,ceil(N/r)) > p, [r 1]);
    prbs = prbs(1:N)';

    w = sigma(1+prbs) .* randn(N,1);
end
```

A.2. Optimality Testing

A.2.1. Connected Piecewise-Linear Regression (Polyline fitting)

Although combinatorial methods for changepoint pursuit do not scale well, for small problems they remain tractable and can be used to evaluate the magnitude of error introduced by search truncation heuristics. Listing A.5 implements such a method.

LISTING A.5. Exact Search for Best-Fit Polyline

```
function [bestcost, bestx, besty, complexity] =
    exhaustive_polyline_fit_branch_and_bound(y, N, givenx)
    bestcost = inf;
    bestx = [];
    besty = [];

    complexity = 0;

    if nargin < 3
        givenx = 1;
    end;

    x = zeros(N+1,1);
    x(1) = 1;
    x(end) = numel(y);

```

```
    S = zeros(1,N);
    R = S;
    n = S;

    lhs = zeros(N+1, N+1);
    lhs(1,1) = 1;

    recurse(0);

    % Recursion is used to achieve the N-1 nested for loops
    % Depth-first traversal also allows reuse of partial sums
    function recurse(ii)
        if ii > 0
            trange = x(ii)+(ii>1):x(ii+1);
            yrange = y(trange);
            complexity = complexity + numel(yrange);
            n(ii) = x(ii+1) - x(ii);
            S(ii) = sum(yrange);
            R(ii) = (trange - x(ii)) * yrange / n(ii);
            a11 = (n(ii)-1)*(2*n(ii)-1);
            a12 = (n(ii)-1)*(n(ii)+1);
            a22 = (n(ii)+1)*(2*n(ii)+1);
            A = [ a11, a12; a12, a22 ]/6/n(ii);
            lhs(ii:ii+1,ii:ii+1) = lhs(ii:ii+1,ii:ii+1) + A;

            b = lhs(1:ii+1,1:ii+1) \ ([S(1:ii)' - R(1:ii)'; 0] + [0; R(1:ii)']);

            yest = interp1q(x(1:ii+1), b, (x(1):x(ii+1))');
            prefix_cost = norm(y(x(1):x(ii+1)) - yest,2);

            if prefix_cost > bestcost
                return;
            end

            if ii == N
                bestcost = prefix_cost;
                bestx = x;
                besty = b;
                return;
            end
        end

        if ii == N-1
            recurse(N);
        elseif ii+2 <= numel(givenx)
            % allows evaluation using predetermined changepoints instead of
            % performing search
```

```
            x(ii+2) = givenx(ii+2);
            recurse(ii+1);
        else
            lhsbefore = lhs;
            for jj = x(ii+1):x(end)-N+ii
                x(ii+2) = jj+1;
                recurse(ii+1);
                lhs = lhsbefore;
            end
        end
    end
end
```

A.3. VIVA Implementation

A.3.1. Piecewise-Constant State Update and Branch

Listing A.6. Piecewise-Constant State Update and Branch

```
using System;

namespace viva.PiecewiseConstant
{
   using Coordinate = System.Collections.Generic.KeyValuePair<uint, double>;

   public class AperiodicState : IAperiodicState<AperiodicState>
   {
      uint j;
      double n;
      double t;
      double S;
      double S2;
      double prior_residual, total_residual;
      double max_deficit;
      ImmutableList<Coordinate> hist;

      public AperiodicState()
      {
      }

      public void Init(SampleData s0)
      {
         n = s0.weight;
         t = s0.t;
         S = s0.weightedValue;
         S2 = s0.weightedSquared;
```

```
28          /* these are all the default state
29             prior_residual = 0.0;
30             hist = null;
31             max_deficit = 0.0;
32           * */
33       }
34
35       public AperiodicState(SolverTags.StepTypeContinue unused,
36                             AperiodicState state, SampleData sk)
37       {
38          j = state.j;
39          t = state.t;
40          n = state.n + sk.weight;
41          S = state.S + sk.weightedValue;
42          S2 = state.S2 + sk.weightedSquared;
43
44          prior_residual = state.prior_residual;
45          max_deficit = state.max_deficit;
46
47          hist = state.hist;
48
49          total_residual = prior_residual + S2 - S * S / n;
50       }
51
52       public AperiodicState Continue(SampleData sk)
53       {
54          return new AperiodicState(SolverTags.Continue, this, sk);
55       }
56
57       public AperiodicState(SolverTags.StepTypeBranch unused,
58                             AperiodicState state,
59                             uint k, double tk, double branch_cost)
60       {
61          j = k;
62          t = tk;
63          n = 0;
64          S = 0.0;
65          S2 = 0.0;
66
67          prior_residual = state.total_residual + branch_cost;
68          total_residual = prior_residual;
69          max_deficit = state.max_deficit;
70
71          hist = new Coordinate(state.j, state.S / state.n).Cons(state.hist);
72       }
73
74       public AperiodicState Branch(uint k, double tk, double branch_cost)
```

```
75      {
76         return
77            new AperiodicState(SolverTags.Branch, this, k, tk, branch_cost);
78      }
79
80      public Func<AperiodicState, bool> Dominates(double branch_cost)
81      {
82         return delegate (AperiodicState candidate)
83         {
84            return candidate.TotalResidual
85                     >= this.TotalResidual + branch_cost;
86         };
87      }
88
89      public Func<AperiodicState, bool> DominatesIfBranch(double branch_cost)
90      {
91         return delegate (AperiodicState candidate)
92         {
93            return candidate.TotalResidual > this.TotalResidual;
94         };
95      }
96
97      public void MarkDeficit(double best_residual)
98      {
99         if (total_residual > best_residual + max_deficit)
100           max_deficit = total_residual - best_residual;
101     }
102     public double MaxDeficit { get { return max_deficit; } }
103
104     public uint PathLength { get { if (hist == null) return 0;
105                                    return hist.Length + 1; } }
106
107     public double TotalResidual{ get { return total_residual; } }
108     public uint SegmentEdge { get { return j; } }
109     public Coordinate[] Path(uint k, double tk)
110     {
111        return new Coordinate(j, S/n).Cons(hist).ToArray();
112     }
113
114     public double[] Estimate(double t)
115     {
116        if (t > this.t && n > 0)
117           return new[] { S / n };
118
119        for( var segment = hist; segment != null; segment = segment.pred ) {
120           if (t > segment.Value.Key)
121              return new[] { segment.Value.Value };
```

```
122         }
123
124         return new [] { double.NaN };
125       }
126    }
127 }
```

A.3.2. Piecewise-Linear State Update and Branch

Listing A.7. Piecewise-Linear State Update and Branch

```
1  using System;
2
3  namespace viva.PiecewiseLinear
4  {
5     using Coordinate = System.Collections.Generic.KeyValuePair<uint, double>;
6
7     public class AperiodicState : IAperiodicState<AperiodicState>
8     {
9        internal struct Summary
10       {
11          public int histID;
12          public uint j2;
13          public float chi2, nu;
14       }
15
16       uint j1, j2;
17       double n12, n23;
18       double t1, t2;
19       double T_12, T_23;
20       double T2_12, T2_23;
21       double S_12, S_23;
22       double S2_12, S2_23;
23       double R_12, R_23;
24       double prior_residual, total_residual;
25       double chi1, chi2, nu;
26       double max_deficit;
27       bool would_prune;
28       ImmutableList<Coordinate> hist;
29
30       public AperiodicState()
31       {
32       }
33
34       public void Init(SampleData s0)
35       {
36          j1 = j2 = s0.k;
37          n23 = s0.weight;
```

```
        t1 = t2 = s0.t;
        S_23 = s0.weightedValue;
        S2_23 = s0.weightedSquared;
        /* these are all the default state
           n12 = 0.0;
           T_12 = T_23 = 0.0;
           T2_12 = T2_23 = 0.0;
           S_12 = 0.0;
           S2_12 = 0.0;
           R_12 = R_23 = 0.0;
           chi1 = 0.0;
           prior_residual = 0.0;
           hist = null;
           chi2 = nu = 0.0;
           max_deficit = 0.0;
           would_prune = false;
         * */
    }

    public AperiodicState(SolverTags.StepTypeContinue unused,
                          AperiodicState state, SampleData sk)
    {
        j1 = state.j1;
        j2 = state.j2;
        t1 = state.t1;
        t2 = state.t2;
        double t23 = sk.t - t2;
        double t13 = sk.t - t1;
        n12 = state.n12;
        n23 = state.n23 + sk.weight;
        T_12 = state.T_12;
        T_23 = state.T_23 + t23 * sk.weight;
        T2_12 = state.T2_12;
        T2_23 = state.T2_23 + t23 * t23 * sk.weight;
        S_12 = state.S_12;
        S_23 = state.S_23 + sk.weightedValue;
        S2_12 = state.S2_12;
        S2_23 = state.S2_23 + sk.weightedSquared;
        R_12 = state.R_12;
        R_23 = state.R_23 + t23 * sk.weightedValue;

        chi1 = state.chi1;
        prior_residual = state.prior_residual;
        max_deficit = state.max_deficit;
        would_prune = state.would_prune;

        hist = state.hist;
```

```
85
86          SolveHelper();
87       }
88
89       public AperiodicState Continue(SampleData sk)
90       {
91          return new AperiodicState(SolverTags.Continue, this, sk);
92       }
93
94       private void SolveHelper()
95       {
96          double t12 = t2 - t1;
97          double terma = 0.0;
98          if (n12 > 0)
99             terma += T2_12 / t12 / t12;
100         double a11 = terma + n23;
101         double a12 = T_23;
102         double a22 = T2_23;
103
104         double b1 = chi1 * terma + S_23;
105         if (n12 > 0)
106         {
107            b1 -= chi1 * T_12 / t12;
108            b1 += R_12 / t12;
109         }
110         double b2 = R_23;
111
112         double det = (a11 * a22 - a12 * a12);
113         if (Math.Abs(det) > 1e-15)
114         {
115            double idet = 1.0 / det;
116            chi2 = (a22 * b1 - a12 * b2) * idet;
117            nu = (a11 * b2 - a12 * b1) * idet;
118         }
119         else
120         {
121            chi2 = S_23 / n23;
122            nu = 0;
123         }
124
125         double residual_term12 = Residual12();
126         double residual_term23 = Residual23();
127         // these terms can't be negative, but
128         // sometimes due to rounding error the computations are
129         total_residual = prior_residual + Math.Max(0, residual_term12)
130                                         + Math.Max(0, residual_term23);
131      }
```

```
132
133       private double Residual12()
134       {
135          if (n12 <= 0)
136             return 0.0;
137
138          double termb = (chi2 - chi1) / (t2 - t1);
139          return S2_12
140             - 2 * chi1 * S_12
141             + chi1 * chi1 * n12
142             + 2 * chi1 * termb * T_12
143             + termb * termb * T2_12
144             - 2 * termb * R_12;
145       }
146
147       private double Residual23()
148       {
149          double result = S2_23 - 2 * chi2 * S_23
150                          + chi2 * chi2 * n23 - nu * nu * T2_23;
151          return Math.Max(0, result);
152       }
153
154       public AperiodicState(SolverTags.StepTypeBranch unused,
155                             AperiodicState state,
156                             uint k, double tk, double branch_cost)
157       {
158          j1 = state.j2;
159          j2 = k;
160          t1 = state.t2;
161          t2 = tk;
162          n12 = state.n23;
163          n23 = 0;
164          T_12 = state.T_23;
165          T_23 = 0.0;
166          T2_12 = state.T2_23;
167          T_23 = 0.0;
168          S_12 = state.S_23;
169          S_23 = 0.0;
170          S2_12 = state.S2_23;
171          S2_23 = 0.0;
172          R_12 = state.R_23;
173          R_23 = 0.0;
174          chi1 = state.chi2;
175          chi2 = state.chi2 + state.nu * (t2 - t1);
176
177          prior_residual = state.prior_residual + branch_cost
178                             + Math.Max(0,state.Residual12());
```

```
179         total_residual = state.total_residual + branch_cost;
180         max_deficit = state.max_deficit;
181         would_prune = state.would_prune;
182
183         hist = new Coordinate(j1, chi1).Cons(state.hist);
184      }
185
186      public AperiodicState Branch(uint k, double t, double branch_cost)
187      {
188         return
189            new AperiodicState(SolverTags.Branch, this, k, t, branch_cost);
190      }
191
192      public Func<AperiodicState, bool> Dominates(double branch_cost)
193      {
194         if (would_prune) Program.ReportFalsePrune();
195         return delegate (AperiodicState candidate)
196         {
197            if (candidate.TotalResidual >= this.TotalResidual + 2 * branch_cost)
198               return true;
199            return false;
200         };
201      }
202      public Func<AperiodicState, bool> DominatesIfBranch(double branch_cost)
203      {
204         return delegate(AperiodicState candidate)
205         {
206            if (candidate.TotalResidual >
207                     this.TotalResidual + branch_cost) return true;
208            return false;
209         };
210      }
211
212      public void MarkDeficit(double best_residual)
213      {
214         if (total_residual > best_residual + max_deficit)
215            max_deficit = total_residual - best_residual;
216      }
217      public double MaxDeficit { get { return max_deficit; } }
218
219      public uint PathLength { get { if (hist == null) return 0;
220                                     return hist.Length + 2; } }
221
222      public double TotalResidual { get { return total_residual; } }
223      public uint SegmentEdge { get { return j2; } }
224      public Coordinate[] Path(uint k, double tk)
225      {
```

```
226         if (would_prune) Program.ReportFalsePruneOptimum();
227         return new Coordinate(k, chi2 + nu * (tk - t2))
228                               .Cons(new Coordinate(j2, chi2)
229                               .Cons(hist)).ToArray();
230      }
231
232      internal void MarkPotentialPrune()
233      {
234         would_prune = true;
235      }
236
237      public double[] Estimate(double t)
238      {
239         if (t > t2)
240            return new[] { chi2 + nu * (t - t2), nu };
241
242         double chileft = chi1;
243         double chiright = chi2;
244         double tleft = t1;
245         double tright = t2;
246         var segment = hist;
247         while (true) {
248            double slope = (chiright - chileft) / (tright - tleft);
249            if (t > tleft)
250               return new [] { chileft + slope * (t - tleft), slope };
251
252            if (segment == null)
253               return new[] { double.NaN, double.NaN };
254
255            chiright = chileft;
256            tright = tleft;
257            chileft = segment.Value.Value;
258            tleft = segment.Value.Key;
259            segment = segment.pred;
260         }
261      }
262
263      internal ImmutableList<Coordinate> GetResults(ref Summary s)
264      {
265         s.j2 = j2;
266         s.chi2 = (float)chi2;
267         s.nu = (float)nu;
268         return hist;
269      }
270   }
271}
```

A.3.3. Pruned Depth-First Search

Listing A.8. Pruned Depth-First Search

```
using System;
using System.Collections.Generic;
using System.Linq;

using System.Threading.Tasks;

namespace viva
{
   using CancellationToken = System.Threading.CancellationToken;

   static class ExtensionMethods
   {
      public static T Minimizer<T>(this IEnumerable<T> seq,
                                   Func<T, double> selector)
      {
         var best = seq.First();
         var best_value = selector(best);

         foreach (T other in seq.Skip(1))
         {
            var other_value = selector(other);
            if (other_value < best_value)
            {
               best_value = other_value;
               best = other;
            }
         }

         return best;
      }
   }

   public class Solvers
   {
      public double StepPenalty = 25.0;
      public int StepInterval = 1;
      public IOnlineEstimatesSink online;
      public uint[] track;

      public enum Branching
      {
         PruneOnly,
         MaximumOneBranch,
         OneBranchPerPivot
```

```
45      }
46
47      public Branching BranchingPolicy = Branching.MaximumOneBranch;
48
49      public enum Weighting
50      {
51         Uniform,
52         Autocorrelation,
53      }
54
55      public Weighting WeightingPolicy = Weighting.Uniform;
56
57      public int TimeoutMilliseconds = 100000;
58
59      protected async Task<KeyValuePair<uint, double>[]> Iteration<tState>(
60            Action<SolverProgressEventArgs<tState>> cb,
61            Func<SampleData, Task<bool>> sampler)
62         where tState : class, IAperiodicState<tState>, new()
63         {
64            var pop = new List<tState>();
65            tState tracking = new tState();
66            SampleData sample = new SampleData();
67            await sampler(sample);
68            tracking.Init(sample);
69            pop.Add(tracking);
70            int iTrack = 1;
71            int rankTrack = 0;
72
73            int largestPopulation = 1;
74
75            var best = tracking;
76            if (online != null)
77               foreach (float tdelay in online.Delays) {
78                  online.AppendEstimates(best.Estimate(sample.t - tdelay));
79               }
80
81            var elapsed = new System.Diagnostics.Stopwatch();
82            while (await sampler(sample))
83            {
84               elapsed.Restart();
85               var next_pop = new List<tState>();
86               next_pop.Capacity = pop.Count + 1;
87
88               var step_best = best.Continue(sample);
89               var best_residual = step_best.TotalResidual;
90               var tracking_residual = 0.0;
91               tState nextTracking = null;
```

```
92
93              tState c;
94              var filter_all = step_best.Dominates(StepPenalty);
95              foreach (var item in pop) {
96                 if (filter_all(item)) continue;
97                 c = item.Continue(sample);
98                 if (filter_all(c)) continue;
99                 next_pop.Add(c);
100                c.MarkDeficit(best_residual);
101
102                if (ReferenceEquals(item, tracking)) nextTracking = c;
103             }
104             best = next_pop.Minimizer(a => a.TotalResidual);
105             if ((sample.k + 1) % StepInterval == 0)
106             {
107                switch (BranchingPolicy) {
108                   case Branching.MaximumOneBranch: {
109                      next_pop.Add(c = best.Branch(sample.k,
110                                                   sample.t,
111                                                   StepPenalty));
112                      c.MarkDeficit(best_residual);
113
114                      if (track != null && sample.k == track[iTrack]) {
115                         nextTracking = c;
116                         iTrack++;
117                      }
118                   }
119                   break;
120                   case Branching.OneBranchPerPivot: {
121                      int[] best_by_changepoint = new int[sample.k + 1];
122                      int[] loc_by_changepoint = new int[sample.k + 1];
123                      var continue_pop = next_pop;
124                      next_pop = new List<tState>();
125                      next_pop.Capacity = continue_pop.Count * 2;
126                      var filter_branches = best.DominatesIfBranch(StepPenalty);
127                      int continue_index = 0;
128                      foreach (var item in continue_pop) {
129                         if (elapsed.ElapsedMilliseconds > TimeoutMilliseconds)
130                            throw new TimeoutException();
131                         ++continue_index;
132                         next_pop.Add(item);
133                         if (filter_branches(item)) continue;
134
135                         int existing = best_by_changepoint[item.SegmentEdge];
136                         if (existing != 0) {
137                            if (continue_pop[existing - 1].TotalResidual
138                                                   < item.TotalResidual)
```

```
                        continue;
                    }
                    c = item.Branch(sample.k, sample.t, StepPenalty);
                    if (existing != 0) {
                       next_pop[loc_by_changepoint[item.SegmentEdge]] = c;
                    }
                    else {
                       loc_by_changepoint[item.SegmentEdge] = next_pop.
                           Count;
                       next_pop.Add(c);
                    }
                    best_by_changepoint[item.SegmentEdge] = continue_index;
                 }
              }
              break;
              case Branching.PruneOnly: {
                 var continue_pop = next_pop;
                 next_pop = new List<tState>();
                 next_pop.Capacity = continue_pop.Count * 2;
                 var filter_branches = best.DominatesIfBranch(StepPenalty);
                 foreach (var item in continue_pop) {
                    if (elapsed.ElapsedMilliseconds > TimeoutMilliseconds)
                       throw new TimeoutException();
                    next_pop.Add(item);
                    if (filter_branches(item)) continue;
                    c = item.Branch(sample.k, sample.t, StepPenalty);
                    next_pop.Add(c);
                 }
              }
              break;
           }
        }

        pop = next_pop;
        if (pop.Count > largestPopulation)
           largestPopulation = pop.Count;

        if (online != null)
           foreach( float tdelay in online.Delays )
           {
              online.AppendEstimates(best.Estimate(sample.t - tdelay));
           }

        tracking = nextTracking;
        if (track != null && tracking != null) {
           tracking_residual = tracking.TotalResidual;
           rankTrack = pop.Count(item =>
```

```
                                        item.TotalResidual < tracking_residual);
            }

            if (cb != null)
               cb(new SolverProgressEventArgs<tState> {
                     Leader = best,
                     Step = sample.k,
                     Time = sample.t,
                     PopulationSize = pop.Count,
                     PruningStandard = best_residual,
                     Tracked = tracking,
                     TrackedRank = rankTrack,
                     TrackedResidual = tracking_residual,
                     PathLength = best.PathLength
                  });
         }

         System.Diagnostics.Trace.WriteLine(
            "final cost = " + best.TotalResidual.ToString());
         System.Diagnostics.Trace.WriteLine(
            "max deficit = " + best.MaxDeficit.ToString());
         System.Diagnostics.Trace.WriteLine(
            "largest candidate population = " + largestPopulation.ToString());
         return best.Path(sample.k, sample.t);
      }
   }

   public class ArraySolvers : Solvers
   {
      public ArraySegment<float> y, t;
      public CancellationToken cancel;

      private Func<SampleData, Task<bool>> BuildSampler()
      {
         CancellationToken cancel = this.cancel;
         switch (WeightingPolicy) {
            case Weighting.Uniform: {
               uint k = 0;
               return async delegate(SampleData result)
               {
                  cancel.ThrowIfCancellationRequested();

                  if (k >= y.Count) return false;
                  var t_k = t.Array[t.Offset + k];
                  var y_k = y.Array[y.Offset + k];
                  result.k = k++;
                  result.t = t_k;
```

```
            result.y = y_k;
            result.weight = 1.0;
            result.weightedValue = y_k;
            result.weightedSquared = y_k * y_k;
            return true;
         };
      }

      case Weighting.Autocorrelation: {
         uint k = 0;

         int support = 10;
         int effective_support = 0;
         double isupp = 1.0;

         double sum = 0, energy = 0;
         int j = 0;
         for (; j < support; ++j) {
            var y_j = y.Array[y.Offset + j];
            sum += y_j;
            energy += y_j * y_j;
            effective_support++;
         }

         return async delegate(SampleData result)
         {
            cancel.ThrowIfCancellationRequested();

            if (k >= y.Count) return false;

            if (k < support) {
               effective_support++;
               isupp = 1.0 / effective_support;
            }
            else {
               var y_jout = y.Array[y.Offset + k - support];
               sum -= y_jout;
               energy -= y_jout * y_jout;
            }
            if (k >= y.Count - support) {
               effective_support--;
               isupp = 1.0 / effective_support;
            }
            else {
               var y_jin = y.Array[y.Offset + k + support];
               sum += y_jin;
               energy += y_jin * y_jin;
```

```
279                 }
280                 var local_var = (energy - sum * sum * isupp) * isupp;
281                 var confidence = 1.0 / (1.0 + local_var);
282
283                 var y_k = y.Array[y.Offset + k];
284                 var t_k = t.Array[t.Offset + k];
285
286                 result.k = k++;
287                 result.t = t_k;
288                 result.y = y_k;
289                 result.weight = confidence;
290                 result.weightedValue = y_k * confidence;
291                 result.weightedSquared = y_k * y_k * confidence;
292                 return true;
293              };
294           }
295           default:
296              throw new InvalidOperationException("Invalid WeightingPolicy");
297        }
298     }
299
300     public KeyValuePair<uint, double>[] OptimizeAperiodic<tState>(
301           Action<SolverProgressEventArgs<tState>> cb)
302        where tState : class, IAperiodicState<tState>, new()
303        {
304           return Iteration(cb, BuildSampler()).Result;
305        }
306
307     public KeyValuePair<uint, double>[] OptimizeAperiodic<tState>()
308        where tState : class, IAperiodicState<tState>, new()
309        {
310           return OptimizeAperiodic<tState>(delegate { });
311        }
312  }
313 }
```

APPENDIX B

Solutions to Least-Squares Subproblems

As VIVA performs basis set pursuit via pruned breadth-first search, computing the cost of each candidate path requires solving a least-squares optimization problem with fixed timing.

Derivations for the least-squares solution and associated integral-square residual error are given for each of the subproblems encountered. These least-squares optimization algorithms can also be used when the changepoint timings are provided by another method, such as brute-force search, convex ℓ^1 estimation, or from the "transmitter" of a synthesized signal used in Monte Carlo simulation.

Additional notes and derivations covering startup, continuous-time, and alternate model formulations (change-of-variables) may be obtained at the companion website:

`http://www.sparxeng.com/viva/`

B.1. Minimum-Residual Constant Segment Estimate

The most complex case is that of arbitrary timing and weights. The general result may be adapted to the simpler models, for example by setting $w_i = 1.0$ everywhere to achieve uniform weighting, or $t_n = nT_s$ to achieve periodic sampling without erasure.

The system is a piecewise-constant signal corrupted by additive white Gaussian noise during measurement. Due to separability of the disjoint curve, it is sufficient to consider a single segment of the signal, having constant value X_k and bounded in time.

$$x(t) = X_k \quad \forall \mathcal{T}_k \le t < \mathcal{T}_{k+1} \tag{B.1a}$$

$$y(t) = x(t) + w(t) \tag{B.1b}$$

$$w \sim \mathcal{N}(0, \sigma^2) \tag{B.1c}$$

B.1.1. Least-Squares Solution

The general integral-square residual cost function for an estimator χ of the original signal X_k is given by

$$\begin{aligned} \epsilon_k(\chi) &= \sum_{\substack{i \\ \mathcal{T}_k < t_i \le \mathcal{T}_{k+1}}} w_i \left(x(t_i) - y_i\right)^2 \\ &= \sum_{\substack{i \\ \mathcal{T}_k < t_i \le \mathcal{T}_{k+1}}} w_i \left(\chi - y_i\right)^2 \end{aligned} \tag{B.2a}$$

This quadratic function is convex, therefore there is a unique critical point, the minimizer $\tilde{\chi}_k$.

$$\frac{d}{d\chi}\epsilon_k(\chi)\bigg|_{\chi=\tilde{\chi}_k} = 0 \tag{B.3a}$$

$$\sum_{\substack{i \\ \mathcal{T}_k<t_i\leq\mathcal{T}_{k+1}}} \frac{d}{d\chi} w_i \left(\chi - y_i\right)^2\bigg|_{\chi=\tilde{\chi}_k} = 0 \tag{B.3b}$$

$$2 \sum_{\substack{i \\ \mathcal{T}_k<t_i\leq\mathcal{T}_{k+1}}} w_i \left(\tilde{\chi}_k - y_i\right) = 0 \tag{B.3c}$$

$$\sum_{\substack{i \\ \mathcal{T}_k<t_i\leq\mathcal{T}_{k+1}}} w_i\tilde{\chi}_k - \sum_{\substack{i \\ \mathcal{T}_k<t_i\leq\mathcal{T}_{k+1}}} w_i y_i = 0 \tag{B.3d}$$

$$\tilde{\chi}_k \sum_{\substack{i \\ \mathcal{T}_k<t_i\leq\mathcal{T}_{k+1}}} w_i = \sum_{\substack{i \\ \mathcal{T}_k<t_i\leq\mathcal{T}_{k+1}}} w_i y_i \tag{B.3e}$$

$$\tilde{\chi}_k = \frac{\sum\limits_{\substack{i \\ \mathcal{T}_k<t_i\leq\mathcal{T}_{k+1}}} w_i y_i}{\sum\limits_{\substack{i \\ \mathcal{T}_k<t_i\leq\mathcal{T}_{k+1}}} w_i} \tag{B.3f}$$

Writing this in terms of state variables (iterative update being described in), a simple expression is obtained:

$$\tilde{\chi}_k = \frac{S_w(\mathcal{T}_k, \mathcal{T}_{k+1})}{N_w(\mathcal{T}_k, \mathcal{T}_{k+1})} \tag{B.4a}$$

where

$$N_c(\mathcal{T}_a, \mathcal{T}_b) = \sum_{\substack{i \\ \mathcal{T}_a<t_i\leq\mathcal{T}_b}} c_i \tag{B.4b}$$

and

$$S_c(\mathcal{T}_a, \mathcal{T}_b) = \sum_{\substack{i \\ \mathcal{T}_a<t_i\leq\mathcal{T}_b}} c_i y_i \tag{B.4c}$$

B.1.2. Residual Error

Residual integral-square error appears in the bicriterion metric being minimized as well as the dynamic programming pruning test (Equations 3.8 and 3.11). In fact, the actual least-squares minimizer only needs to be calculated for the optimal changepoint timing, while the minimum error term needs to be calculated for every candidate. Therefore a direct expression for residual error is desirable.

$$\begin{aligned}
\epsilon_k(\tilde{\chi}_k) &= \sum_{\substack{i \\ \mathcal{T}_k<t_i\leq\mathcal{T}_{k+1}}} w_i \left(\tilde{\chi}_k - y_i\right)^2 \\
&= \sum_{\substack{i \\ \mathcal{T}_k<t_i\leq\mathcal{T}_{k+1}}} w_i \left(\tilde{\chi}_k - y_i\right)\tilde{\chi}_k - \sum_{\substack{i \\ \mathcal{T}_k<t_i\leq\mathcal{T}_{k+1}}} w_i \left(\tilde{\chi}_k - y_i\right) y_i \\
&= \tilde{\chi}_k \sum_{\substack{i \\ \mathcal{T}_k<t_i\leq\mathcal{T}_{k+1}}} w_i \left(\tilde{\chi}_k - y_i\right) - \tilde{\chi}_k \sum_{\substack{i \\ \mathcal{T}_k<t_i\leq\mathcal{T}_{k+1}}} w_i y_i + \sum_{\substack{i \\ \mathcal{T}_k<t_i\leq\mathcal{T}_{k+1}}} w_i y_i^2
\end{aligned} \tag{B.5a}$$

but the first term is zero from Equation B.3c and therefore

$$\epsilon_k(\tilde{\chi}_k) = \sum_{\substack{i \\ \mathcal{T}_k<t_i\leq\mathcal{T}_{k+1}}} w_i y_i^2 - \tilde{\chi}_k \sum_{\substack{i \\ \mathcal{T}_k<t_i\leq\mathcal{T}_{k+1}}} w_i y_i \tag{B.5b}$$

When expressed in terms of the state variables, this becomes:

$$\epsilon_k(\tilde{\chi}_k) = S_w^2(\mathcal{T}_k, \mathcal{T}_{k+1}) - \frac{S_w(\mathcal{T}_k, \mathcal{T}_{k+1})^2}{N_w(\mathcal{T}_k, \mathcal{T}_{k+1})} \tag{B.6a}$$

where

$$S_c^p(\mathcal{T}_a, \mathcal{T}_b) = \sum_{\substack{i \\ \mathcal{T}_a < t_i \leq \mathcal{T}_b}} c_i y_i^p \tag{B.6b}$$

Note that $S_c^p(\mathcal{T}_k, \mathcal{T}_{k+1}) \neq S_c(\mathcal{T}_k, \mathcal{T}_{k+1})^p$ in general

B.2. Minimum-Residual Polyline Estimate

B.2.1. Least-Squares Solution

In the case of polyline curve fitting (conjoined piecewise-linear segments), each estimated point is an affine combination of the two estimated vertices which bracket it.

$$x(t) = \frac{\chi_k \left(\mathcal{T}_{k+1} - t\right) + \chi_{k+1} \left(t - \mathcal{T}_k\right)}{\mathcal{T}_{k+1} - \mathcal{T}_k} \quad \forall k : \mathcal{T}_k \leq t \leq \mathcal{T}_{k+1} \tag{B.7}$$

As before, the problem is solved for arbitrary timing and weights, obtaining a general result which may be adapted to simpler cases.

The general integral-square residual cost function may be grouped into terms corresponding to each segment:

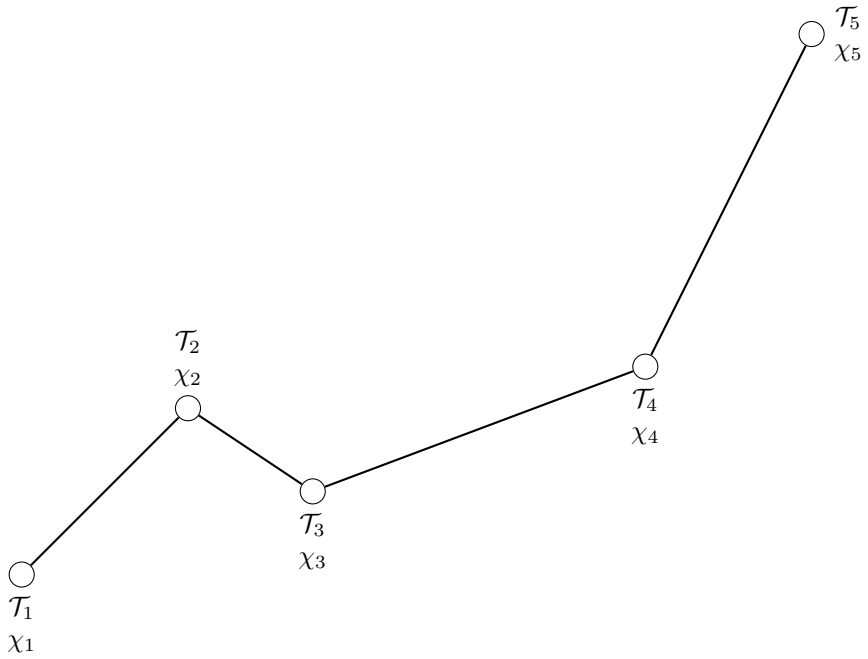


Figure B.1. Fit using many conjoined piecewise-linear segments

$$\begin{aligned}\epsilon(\chi) &= \sum_i w_i \left(x(t_i) - y_i\right)^2 \\ &= w_1 \left(\chi_1 - y_1\right)^2 + \sum_{k=1}^{N-1} \epsilon_k(\chi)\end{aligned} \tag{B.8a}$$

where

$$\begin{aligned}\epsilon_k(\chi) &= \sum_{\substack{i \\ \mathcal{T}_k < t_i \leq \mathcal{T}_{k+1}}} w_i \left(x(t_i) - y_i\right)^2 \\ &= \sum_{\substack{i \\ \mathcal{T}_k < t_i \leq \mathcal{T}_{k+1}}} w_i \left(\frac{\chi_k(\mathcal{T}_{k+1} - t_i) + \chi_{k+1}(t_i - \mathcal{T}_k)}{\mathcal{T}_{k+1} - \mathcal{T}_k} - y_i \right)^2\end{aligned} \tag{B.8b}$$

These terms have partial derivatives

$$\frac{1}{\cancel{2}} \frac{\partial}{\partial \chi_k} \epsilon_k = \frac{\cancel{1}}{\cancel{2}} \sum_{\substack{i \\ \mathcal{T}_k < t_i \leq \mathcal{T}_{k+1}}} \cancel{2} w_i \left(x(t_i) - y_i\right) \frac{\partial}{\partial \chi_k} \left(x(t_i) - y_i\right) \tag{B.9a}$$

$$\begin{aligned}&= \sum_{\substack{i \\ \mathcal{T}_k < t_i \leq \mathcal{T}_{k+1}}} w_i (x(t_i) - y_i) \frac{\mathcal{T}_{k+1} - t_i}{\mathcal{T}_{k+1} - \mathcal{T}_k} \\ &= \frac{\sum_{\substack{i \\ \mathcal{T}_k < t_i \leq \mathcal{T}_{k+1}}} w_i x(t_i)(\mathcal{T}_{k+1} - t_i) - \sum_{\substack{i \\ \mathcal{T}_k < t_i \leq \mathcal{T}_{k+1}}} w_i y_i (\mathcal{T}_{k+1} - t_i)}{\mathcal{T}_{k+1} - \mathcal{T}_k}\end{aligned}$$

but

$$\sum_{\substack{i \\ \mathcal{T}_k < t_i \leq \mathcal{T}_{k+1}}} w_i (t_i - \mathcal{T}_k)(\mathcal{T}_{k+1} - t_i) = \sum_{\substack{i \\ \mathcal{T}_k < t_i \leq \mathcal{T}_{k+1}}} w_i (t_i - \mathcal{T}_k)(\mathcal{T}_{k+1} - \mathcal{T}_k) - \sum_{\substack{i \\ \mathcal{T}_k < t_i \leq \mathcal{T}_{k+1}}} w_i (t_i - \mathcal{T}_k)^2 \tag{B.9b}$$

$$= (\mathcal{T}_{k+1} - \mathcal{T}_k) T_c(\mathcal{T}_k, \mathcal{T}_{k+1}) - T_c^2(\mathcal{T}_k, \mathcal{T}_{k+1})$$

and

(B.9c)

$$\begin{aligned}
\sum_{\substack{i\\ \mathcal{T}_k<t_i\le\mathcal{T}_{k+1}}} w_i(\mathcal{T}_{k+1}-t_i)^2 &= \sum_{\substack{i\\ \mathcal{T}_k<t_i\le\mathcal{T}_{k+1}}} w_i(\mathcal{T}_{k+1}-t_i)(\mathcal{T}_{k+1}-\mathcal{T}_k) - \sum_{\substack{i\\ \mathcal{T}_k<t_i\le\mathcal{T}_{k+1}}} w_i(\mathcal{T}_{k+1}-t_i)(t_i-\mathcal{T}_k)\\
&= (\mathcal{T}_{k+1}-\mathcal{T}_k) \sum_{\substack{i\\ \mathcal{T}_k<t_i\le\mathcal{T}_{k+1}}} w_i(\mathcal{T}_{k+1}-t_i)\\
&\quad - \big((\mathcal{T}_{k+1}-\mathcal{T}_k)T_c(\mathcal{T}_k,\mathcal{T}_{k+1}) - T_c^2(\mathcal{T}_k,\mathcal{T}_{k+1})\big)\\
&= (\mathcal{T}_{k+1}-\mathcal{T}_k) \sum_{\substack{i\\ \mathcal{T}_k<t_i\le\mathcal{T}_{k+1}}} w_i(\mathcal{T}_{k+1}-\mathcal{T}_k)\\
&\quad - (\mathcal{T}_{k+1}-\mathcal{T}_k) \sum_{\substack{i\\ \mathcal{T}_k<t_i\le\mathcal{T}_{k+1}}} w_i(t_i-\mathcal{T}_k)\\
&\quad + T_c^2(\mathcal{T}_k,\mathcal{T}_{k+1}) - (\mathcal{T}_{k+1}-\mathcal{T}_k)T_c(\mathcal{T}_k,\mathcal{T}_{k+1})\\
&= (\mathcal{T}_{k+1}-\mathcal{T}_k)^2 \sum_{\substack{i\\ \mathcal{T}_k<t_i\le\mathcal{T}_{k+1}}} w_i - (\mathcal{T}_{k+1}-\mathcal{T}_k)T_c(\mathcal{T}_k,\mathcal{T}_{k+1})\\
&\quad + T_c^2(\mathcal{T}_k,\mathcal{T}_{k+1}) - (\mathcal{T}_{k+1}-\mathcal{T}_k)T_c(\mathcal{T}_k,\mathcal{T}_{k+1})\\
&= (\mathcal{T}_{k+1}-\mathcal{T}_k)^2 N_c(\mathcal{T}_k,\mathcal{T}_{k+1}) + T_c^2(\mathcal{T}_k,\mathcal{T}_{k+1}) - 2(\mathcal{T}_{k+1}-\mathcal{T}_k)T_c(\mathcal{T}_k,\mathcal{T}_{k+1})
\end{aligned}$$

therefore

(B.9d)

$$\begin{aligned}
\sum_{\substack{i\\ \mathcal{T}_k<t_i\le\mathcal{T}_{k+1}}} w_i x(t_i)(\mathcal{T}_{k+1}-t_i) &= \sum_{\substack{i\\ \mathcal{T}_k<t_i\le\mathcal{T}_{k+1}}} w_i \frac{\chi_k\,(\mathcal{T}_{k+1}-t_i) + \chi_{k+1}\,(t_i-\mathcal{T}_k)}{\mathcal{T}_{k+1}-\mathcal{T}_k}(\mathcal{T}_{k+1}-t_i)\\
&= \frac{\chi_k}{\mathcal{T}_{k+1}-\mathcal{T}_k} \sum_{\substack{i\\ \mathcal{T}_k<t_i\le\mathcal{T}_{k+1}}} w_i(\mathcal{T}_{k+1}-t_i)^2\\
&\quad + \frac{\chi_{k+1}}{\mathcal{T}_{k+1}-\mathcal{T}_k} \sum_{\substack{i\\ \mathcal{T}_k<t_i\le\mathcal{T}_{k+1}}} w_i(t_i-\mathcal{T}_k)(\mathcal{T}_{k+1}-t_i)\\
&= \chi_k\left[(\mathcal{T}_{k+1}-\mathcal{T}_k)N_c(\mathcal{T}_k,\mathcal{T}_{k+1}) - 2T_c(\mathcal{T}_k,\mathcal{T}_{k+1}) + \frac{T_c^2(\mathcal{T}_k,\mathcal{T}_{k+1})}{\mathcal{T}_{k+1}-\mathcal{T}_k}\right]\\
&\quad + \chi_{k+1}\left[T_c(\mathcal{T}_k,\mathcal{T}_{k+1}) - \frac{T_c^2(\mathcal{T}_k,\mathcal{T}_{k+1})}{\mathcal{T}_{k+1}-\mathcal{T}_k}\right]
\end{aligned}$$

also

(B.9e)

$$\begin{aligned}
\sum_{\substack{i\\ \mathcal{T}_k<t_i\le\mathcal{T}_{k+1}}} w_i y_i(\mathcal{T}_{k+1}-t_i) &= \sum_{\substack{i\\ \mathcal{T}_k<t_i\le\mathcal{T}_{k+1}}} w_i y_i(\mathcal{T}_{k+1}-\mathcal{T}_k) - \sum_{\substack{i\\ \mathcal{T}_k<t_i\le\mathcal{T}_{k+1}}} w_i y_i(t_i-\mathcal{T}_k)\\
&= (\mathcal{T}_{k+1}-\mathcal{T}_k)S_c(\mathcal{T}_k,\mathcal{T}_{k+1}) - R_c(\mathcal{T}_k,\mathcal{T}_{k+1})
\end{aligned}$$

so that

(B.9f)

$$\begin{aligned}\frac{1}{2}\frac{\partial}{\partial\chi_k}\epsilon_k &= \frac{1}{\mathcal{T}_{k+1}-\mathcal{T}_k}\left(\chi_k\left[(\mathcal{T}_{k+1}-\mathcal{T}_k)N_c(\mathcal{T}_k,\mathcal{T}_{k+1})-2T_c(\mathcal{T}_k,\mathcal{T}_{k+1})+\frac{T_c^2(\mathcal{T}_k,\mathcal{T}_{k+1})}{\mathcal{T}_{k+1}-\mathcal{T}_k}\right]\right.\\ &\qquad\left.+\chi_{k+1}\left[T_c(\mathcal{T}_k,\mathcal{T}_{k+1})-\frac{T_c^2(\mathcal{T}_k,\mathcal{T}_{k+1})}{\mathcal{T}_{k+1}-\mathcal{T}_k}\right]+(\mathcal{T}_{k+1}-\mathcal{T}_k)S_c(\mathcal{T}_k,\mathcal{T}_{k+1})-R_c(\mathcal{T}_k,\mathcal{T}_{k+1})\right)\\ &= \chi_k\left[N_c(\mathcal{T}_k,\mathcal{T}_{k+1})-2\frac{T_c(\mathcal{T}_k,\mathcal{T}_{k+1})}{\mathcal{T}_{k+1}-\mathcal{T}_k}+\frac{T_c^2(\mathcal{T}_k,\mathcal{T}_{k+1})}{(\mathcal{T}_{k+1}-\mathcal{T}_k)^2}\right]\\ &\quad+\chi_{k+1}\left[\frac{T_c(\mathcal{T}_k,\mathcal{T}_{k+1})}{\mathcal{T}_{k+1}-\mathcal{T}_k}-\frac{T_c^2(\mathcal{T}_k,\mathcal{T}_{k+1})}{(\mathcal{T}_{k+1}-\mathcal{T}_k)^2}\right]+S_c(\mathcal{T}_k,\mathcal{T}_{k+1})-\frac{R_c(\mathcal{T}_k,\mathcal{T}_{k+1})}{\mathcal{T}_{k+1}-\mathcal{T}_k}\end{aligned}$$

In similar fashion,

(B.10a)

$$\begin{aligned}\sum_{\substack{i\\ \mathcal{T}_k<t_i\le\mathcal{T}_{k+1}}} w_i x(t_i)(t_i-\mathcal{T}_k) &= \sum_{\substack{i\\ \mathcal{T}_k<t_i\le\mathcal{T}_{k+1}}} w_i\frac{\chi_k(\mathcal{T}_{k+1}-t_i)+\chi_{k+1}(t_i-\mathcal{T}_k)}{\mathcal{T}_{k+1}-\mathcal{T}_k}(t_i-\mathcal{T}_k)\\ &= \frac{\chi_k}{\mathcal{T}_{k+1}-\mathcal{T}_k}\sum_{\substack{i\\ \mathcal{T}_k<t_i\le\mathcal{T}_{k+1}}} w_i(\mathcal{T}_{k+1}-t_i)(t_i-\mathcal{T}_k)\\ &\quad+\frac{\chi_{k+1}}{\mathcal{T}_{k+1}-\mathcal{T}_k}\sum_{\substack{i\\ \mathcal{T}_k<t_i\le\mathcal{T}_{k+1}}} w_i(t_i-\mathcal{T}_k)^2\\ &= \chi_k\left[T_c(\mathcal{T}_k,\mathcal{T}_{k+1})-\frac{T_c^2(\mathcal{T}_k,\mathcal{T}_{k+1})}{\mathcal{T}_{k+1}-\mathcal{T}_k}\right]+\chi_{k+1}\frac{T_c^2(\mathcal{T}_k,\mathcal{T}_{k+1})}{\mathcal{T}_{k+1}-\mathcal{T}_k}\end{aligned}$$

so that

(B.10b)

$$\begin{aligned}\frac{1}{2}\frac{\partial}{\partial\chi_{k+1}}\epsilon_k &= \frac{1}{2}\sum_{\substack{i\\ \mathcal{T}_k<t_i\le\mathcal{T}_{k+1}}} 2w_i(x(t_i)-y_i)\frac{\partial}{\partial\chi_{k+1}}(x(t_i)-y_i)\\ &= \sum_{\substack{i\\ \mathcal{T}_k<t_i\le\mathcal{T}_{k+1}}} w_i(x(t_i)-y_i)\frac{t_i-\mathcal{T}_k}{\mathcal{T}_{k+1}-\mathcal{T}_k}\\ &= \frac{\sum_{\substack{i\\ \mathcal{T}_k<t_i\le\mathcal{T}_{k+1}}} w_i x(t_i)(t_i-\mathcal{T}_k)-\sum_{\substack{i\\ \mathcal{T}_k<t_i\le\mathcal{T}_{k+1}}} w_i y_i(t_i-\mathcal{T}_k)}{\mathcal{T}_{k+1}-\mathcal{T}_k}\\ &= \chi_k\left[\frac{T_c(\mathcal{T}_k,\mathcal{T}_{k+1})}{\mathcal{T}_{k+1}-\mathcal{T}_k}-\frac{T_c^2(\mathcal{T}_k,\mathcal{T}_{k+1})}{(\mathcal{T}_{k+1}-\mathcal{T}_k)^2}\right]\\ &\quad+\chi_{k+1}\frac{T_c^2(\mathcal{T}_k,\mathcal{T}_{k+1})}{(\mathcal{T}_{k+1}-\mathcal{T}_k)^2}-\frac{R_c(\mathcal{T}_k,\mathcal{T}_{k+1})}{\mathcal{T}_{k+1}-\mathcal{T}_k}\end{aligned}$$

The critical point occurs when

$$\nabla_{\chi'}\epsilon(\chi')\big|_{\chi'=\tilde{\chi}} = \mathbf{0} \tag{B.11}$$

which describes a system of linear equations. There is an iterated case (vertex surrounded by two segments) and two base cases for the end vertices.

The first case, which applies only when the first vertex is being optimized, is given by

$$
\begin{aligned}
0 &= \frac{1}{2}\frac{\partial}{\partial \chi'_1}\epsilon(\chi')|_{\chi'=\tilde{\chi}} \\
&= \frac{1}{2}\frac{\partial}{\partial \chi'_1} w_1 \left(\chi'_1 - y_1\right)^2 |_{\chi'=\tilde{\chi}} + \frac{1}{2}\frac{\partial}{\partial \chi'_1}\epsilon_1(\chi')|_{\chi'=\tilde{\chi}} \\
&= \frac{1}{2} 2 w_1(\tilde{\chi}_1 - y_1) + \tilde{\chi}_1 \left[\frac{T_c(\mathcal{T}_1,\mathcal{T}_2)}{\mathcal{T}_2 - \mathcal{T}_1} - \frac{T_c^2(\mathcal{T}_1,\mathcal{T}_2)}{(\mathcal{T}_2 - \mathcal{T}_1)^2}\right] + \tilde{\chi}_2 \frac{T_c^2(\mathcal{T}_1,\mathcal{T}_2)}{(\mathcal{T}_2 - \mathcal{T}_1)^2} - \frac{R_c(\mathcal{T}_1,\mathcal{T}_2)}{\mathcal{T}_2 - \mathcal{T}_1}
\end{aligned} \tag{B.12}
$$

which yields the equation

$$
= \frac{1}{2} 2 w_1(\chi_1 - y_1) + \tilde{\chi}_1 \left[\frac{T_c(\mathcal{T}_1,\mathcal{T}_2)}{\mathcal{T}_2 - \mathcal{T}_1} - \frac{T_c^2(\mathcal{T}_1,\mathcal{T}_2)}{(\mathcal{T}_2 - \mathcal{T}_1)^2}\right] + \tilde{\chi}_2 \frac{T_c^2(\mathcal{T}_1,\mathcal{T}_2)}{(\mathcal{T}_2 - \mathcal{T}_1)^2} - \frac{R_c(\mathcal{T}_1,\mathcal{T}_2)}{\mathcal{T}_2 - \mathcal{T}_1} \tag{B.13}
$$

When the segment-freeze heuristic is being used, this equation is replaced by an explicit value for $\tilde{\chi}_1$.

The final case is

$$
\begin{aligned}
0 &= \frac{1}{2}\frac{\partial}{\partial \chi'_N}\epsilon(\chi')|_{\chi'=\tilde{\chi}} \\
&= \frac{1}{2}\frac{\partial}{\partial \chi'_N}\epsilon_{N-1}(\chi')|_{\chi'=\tilde{\chi}} \\
&= \tilde{\chi}_{N-1}\left[\frac{T_c(\mathcal{T}_{N-1},\mathcal{T}_N)}{\mathcal{T}_N - \mathcal{T}_{N-1}} - \frac{T_c^2(\mathcal{T}_{N-1},\mathcal{T}_N)}{(\mathcal{T}_N - \mathcal{T}_{N-1})^2}\right] + \tilde{\chi}_N \frac{T_c^2(\mathcal{T}_{N-1},\mathcal{T}_N)}{(\mathcal{T}_N - \mathcal{T}_{N-1})^2} - \frac{R_c(\mathcal{T}_{N-1},\mathcal{T}_N)}{\mathcal{T}_N - \mathcal{T}_{N-1}}
\end{aligned} \tag{B.14}
$$

which implies that

$$
\tilde{\chi}_{N-1}\left[\frac{T_c(\mathcal{T}_{N-1},\mathcal{T}_N)}{\mathcal{T}_N - \mathcal{T}_{N-1}} - \frac{T_c^2(\mathcal{T}_{N-1},\mathcal{T}_N)}{(\mathcal{T}_N - \mathcal{T}_{N-1})^2}\right] + \tilde{\chi}_N \frac{T_c^2(\mathcal{T}_{N-1},\mathcal{T}_N)}{(\mathcal{T}_N - \mathcal{T}_{N-1})^2} = \frac{R_c(\mathcal{T}_{N-1},\mathcal{T}_N)}{\mathcal{T}_N - \mathcal{T}_{N-1}} \tag{B.15}
$$

The iterated case is

$$
\begin{aligned}
0 &= \frac{1}{2}\frac{\partial}{\partial \chi'_k}\epsilon(\chi')|_{\chi'=\tilde{\chi}} \\
&= \frac{1}{2}\frac{\partial}{\partial \chi'_k}\epsilon_{k-1}(\chi')|_{\chi'=\tilde{\chi}} + \frac{1}{2}\frac{\partial}{\partial \chi'_k}\epsilon_k(\chi')|_{\chi'=\tilde{\chi}} \\
&= \tilde{\chi}_{k-1}\left[\frac{T_c(t_{k-1},t_k)}{t_k - t_{k-1}} - \frac{T_c^2(t_{k-1},t_k)}{(t_k - t_{k-1})^2}\right] + \tilde{\chi}_k \frac{T_c^2(t_{k-1},t_k)}{(t_k - t_{k-1})^2} - \frac{R_c(t_{k-1},t_k)}{t_k - t_{k-1}} \\
&\quad + \tilde{\chi}_k\left[N_c(t_k,t_{k+1}) - 2\frac{T_c(t_k,t_{k+1})}{t_{k+1} - t_k} + \frac{T_c^2(t_k,t_{k+1})}{(t_{k+1} - t_k)^2}\right] \\
&\quad + \tilde{\chi}_{k+1}\left[\frac{T_c(t_k,t_{k+1})}{t_{k+1} - t_k} - \frac{T_c^2(t_k,t_{k+1})}{(t_{k+1} - t_k)^2}\right] + S_c(t_k,t_k+1)) - \frac{R_c(t_k,t_{k+1})}{t_{k+1} - t_k}
\end{aligned} \tag{B.16}
$$

while implies that

$$
\begin{aligned}
&\tilde{\chi}_{k-1}\left[\frac{T_c(t_{k-1},t_k)}{t_k - t_{k-1}} - \frac{T_c^2(t_{k-1},t_k)}{(t_k - t_{k-1})^2}\right] \\
&\quad + \tilde{\chi}_k\left[\frac{T_c^2(t_{k-1},t_k)}{(t_k - t_{k-1})^2} + N_c(t_k,t_{k+1}) - 2\frac{T_c(t_k,t_{k+1})}{t_{k+1} - t_k} + \frac{T_c^2(t_k,t_{k+1})}{(t_{k+1} - t_k)^2}\right] \\
&\quad + \tilde{\chi}_{k+1}\left[\frac{T_c(t_k,t_{k+1})}{t_{k+1} - t_k} - \frac{T_c^2(t_k,t_{k+1})}{(t_{k+1} - t_k)^2}\right] = \frac{R_c(t_{k-1},t_k)}{t_k - t_{k-1}} - S_c(t_k,t_k+1)) \\
&\quad + \frac{R_c(t_k,t_{k+1})}{t_{k+1} - t_k}
\end{aligned} \tag{B.17}
$$

Equations B.13, B.17 (repeated), and B.15 form a tridiagonal linear system which may be solved for $\tilde{\chi}$, the vertex ordinates of the polyline of best fit.

APPENDIX C

Summary of measurement statistics

C.1. Continuous time

Rather than calculating using the entire sequence of noisy measurements, algorithm implementations need only calculate a few statistics. The continuous-time algorithms use measurement, energy, and correlation integrals:

$$S(t_A, t_B) = \int_{t_A}^{t_B} y(\tau)\, d\tau \tag{C.1a}$$

$$S^2(t_A, t_B) = \int_{t_A}^{t_B} y(\tau)^2\, d\tau \tag{C.1b}$$

$$R(t_A, t_B) = \int_{t_A}^{t_B} (\tau - t_1)\, y(\tau)\, d\tau \tag{C.1c}$$

C.1.1. Recursive update

Often it is desirable to perform parameter estimation during data collection. In this case, a recursive solution is advantageous, because when it exists, recursively using prior computations requires less memory and processing than reevaluating the direct formula.

The following recurrence relationship holds for the continuous-time statistics.

$$S^i(t + \Delta t) = S^i(t) + \int_{t}^{t+\Delta t} y(\tau)^i\, d\tau \tag{C.2a}$$

C.1.2. Change of anchor time

For retrospective analysis, the ability to efficiently recalculate using different timing selections is desired. The following relationships allow simple computation of the statistics from terms which are independent of step time sequences:

$$S^m(t_A, t_B) = S^m(t_0, t_B) - S^m(t_0, t_A) \tag{C.3a}$$

$$R(t_A, t_B) = R(t_0, t_B) - R(t_0, t_A) - (t_A - t_0)\, S(t_A, t_B) \tag{C.3b}$$

Derivation:

$$S^m(t_A, t_B) = \int_{t_A}^{t_B} y(\tau)^m\, d\tau \tag{C.4a}$$

$$= \int_{t_0}^{t_B} y(\tau)^m\, d\tau - \int_{t_0}^{t_A} y(\tau)^m\, d\tau \tag{C.4b}$$

$$R(t_A, t_B) = \int_{t_A}^{t_B} (\tau - t_A)\, y(\tau)\, d\tau \tag{C.5a}$$

$$= \int_{t_A}^{t_B} (\tau - t_0)\, y(\tau)\, d\tau - \int_{t_A}^{t_B} (t_A - t_0)\, y(\tau)\, d\tau \tag{C.5b}$$

$$= \int_{t_0}^{t_B} (\tau - t_0)\, y(\tau)\, d\tau - \int_{t_0}^{t_A} (\tau - t_0)\, y(\tau)\, d\tau - (t_A - t_0) \int_{t_A}^{t_B} y(\tau)\, d\tau \tag{C.5c}$$

C.2. Discrete time

Discrete-time algorithms use measurement, energy, and correlation sums. An optional weighting term c_i was introduced in Equation 4.6.

$$N_c(t_A, t_B) = \sum_{i|t_A<t_i\leq t_B} c_i \tag{C.6a}$$

$$T_c(t_A, t_B) = \sum_{i|t_A<t_i\leq t_B} c_i(t_i - t_A) \tag{C.6b}$$

$$T_c^2(t_A, t_B) = \sum_{i|t_A<t_i\leq t_B} c_i(t_i - t_A)^2 \tag{C.6c}$$

$$S_c(t_A, t_B) = \sum_{i|t_A<t_i\leq t_B} c_i y_i \tag{C.6d}$$

$$S_c^2(t_A, t_B) = \sum_{i|t_A<t_i\leq t_B} c_i y_i^2 \tag{C.6e}$$

$$R_c(t_A, t_B) = \sum_{i|t_A<t_i\leq t_B} c_i(t_i - t_A) y_i \tag{C.6f}$$

C.2.1. Recursive Update

In the periodically sampled case, where $t_n = nT_s$, let

$$(S^i)_n = \frac{S^i(t_n)}{T_s} \tag{C.7a}$$

$$(S^i)_{n+1} = (S^i)_n + y_n^i \tag{C.7b}$$

and

$$\tilde{x}_n = \frac{S_n}{n - n_1} \tag{C.7c}$$

$$\tilde{\epsilon}_n = \frac{(S^2)_n}{n - n_1} - \tilde{x}_n^2 \tag{C.7d}$$

APPENDIX D

Linear Relaxations

Branch-and-bound is a technique for reducing the search space for the optimal solution to an integer, binary, or mixed-integer program from the 2^n distinct quadratic programs implied by the n binary variables $b_1, b_2, \ldots, b_n$.

In particular, the optimal cost of the relaxed program, with having a subset K of the variables b'_k fixed

$$
\begin{array}{lrcl}
\text{minimize} & \hat{c}_K = (\hat{\mathbf{x}} - \mathbf{y})^T (\hat{\mathbf{x}} - \mathbf{y}) + \zeta \mathbf{1}^T \hat{\mathbf{b}} & & \\
\text{subject to} & -\hat{x}_k + \hat{x}_{k+1} = 0 & & \forall k \in \{\mathrm{K} | b'_k = 0\} \\
& -\hat{x}_k + \hat{x}_{k+1} - M\hat{b}_k \leq 0 & & \forall k \notin \mathrm{K} \\
& \hat{x}_k - \hat{x}_{k+1} - M\hat{b}_k \leq 0 & & \forall k \notin \mathrm{K} \\
& \hat{x}_k = x'_k & & \forall k \in \mathrm{K} \\
& \hat{b}_k = b'_k & & \forall k \in \mathrm{K} \\
& 0 \leq \hat{b}_k & & \forall k \notin \mathrm{K}
\end{array}
\tag{D.1}
$$

provides a lower bound for the optimal cost of all binary programs sharing the same values of b'_k for the set K.

$$\hat{c}_K \leq c_K \tag{D.2}$$

Special attention will be given to the causal set

$$\mathrm{K}_{\leq \kappa} = \{k | k \leq \kappa\} \tag{D.3}$$

The causal set is particularly important because it allows online optimization at sample number κ making maximal use of prior samples and, as will be shown, no future information is required to do so.

Consider that each term of the goal function is non-negative.

THEOREM 2. *For finite y_k, if $\mathbf{x'}_{\leq\kappa}, \mathbf{b'}_{\leq\kappa}$ is any solution to*

$$
\begin{array}{lrcl}
\textit{minimize} & c_{\leq\kappa} = \sum_{k=0}^{\kappa} (x'_k - y_k)^2 + \zeta \sum_{k=0}^{\kappa-1} b'_k & & \\
\textit{subject to} & -x'_k + x'_{k+1} - Mb'_k \leq 0 & & \forall k < \kappa \\
& x'_k - x'_{k+1} - Mb'_k \leq 0 & & \forall k < \kappa \\
& b'_k \in \{0, 1\} & & \forall k < \kappa
\end{array}
\tag{D.4}
$$

The choice of

$$\hat{x}_k = \begin{cases} x'_k & k \leq \kappa \\ y_k & k > \kappa \end{cases} \tag{D.5}$$

leads to a solution to the relaxed problem D.1 which is:

a) feasible
b) optimal in the limit as $M \to \infty$

PROOF. The corresponding solution is

$$\hat{b}_k = \frac{1}{M} |y_{k+1} - y_k|, \quad \forall \kappa \leq k < N \tag{D.6}$$

a) Feasibility follows directly from the constraints.

b) Consider the cost function

$$\hat{c}_K = (\hat{\mathbf{x}} - \mathbf{y})^T (\hat{\mathbf{x}} - \mathbf{y}) + \zeta \mathbf{1}^T \mathbf{b}' + \zeta \mathbf{1}^T \hat{\mathbf{b}} \tag{D.7a}$$

which rearranges to

$$\begin{aligned} \hat{c}_K &= \sum_{k=0}^{N} (\hat{x}_k - y_k)^2 + \zeta \sum_{k=0}^{N-1} \hat{b}_k \\ &= \sum_{k=0}^{\kappa} (\hat{x}_k - y_k)^2 + \sum_{k=\kappa+1}^{N} (\hat{x}_k - y_k)^2 + \zeta \sum_{k=0}^{\kappa-1} \hat{b}_k + \zeta \sum_{k=\kappa}^{N-1} \hat{b}_k \\ &= \sum_{k=0}^{\kappa} (x'_k - y_k)^2 + \sum_{k=\kappa+1}^{N} (\hat{x}_k - y_k)^2 + \zeta \sum_{k=0}^{\kappa-1} b'_k + \zeta \sum_{k=\kappa}^{N-1} \hat{b}_k \end{aligned} \tag{D.7b}$$

$$\hat{c}_K = c_{\leq \kappa} + \sum_{k=\kappa+1}^{N} (\hat{x}_k - y_k)^2 + \zeta \sum_{k=\kappa}^{N-1} \hat{b}_k \tag{D.7c}$$

Non-negativity implies

$$\hat{c}_K \geq c_{\leq \kappa} \tag{D.7d}$$

By D.5 and D.6,

$$\tilde{c}_K = c_{\leq \kappa} + \sum_{k=\kappa+1}^{N} (y_k - y_k)^2 + \zeta \sum_{k=\kappa}^{N-1} \frac{1}{M} |y_{k+1} - y_k| \tag{D.7e}$$

Because y_k are finite,

$$\lim_{M \to \infty} \frac{\zeta}{M} \sum_{k=\kappa}^{N-1} |y_{k+1} - y_k| = 0 \tag{D.7f}$$

so that

$$\tilde{c}_K \to c_{\leq \kappa} \tag{D.8}$$

and since the lower bound is reached, optimality is assured. □

APPENDIX E

Small Polyline fitting examples

These figures provide additional examples small enough for brute force solution to accompany the discussion in Sections 3.2 and 3.4.

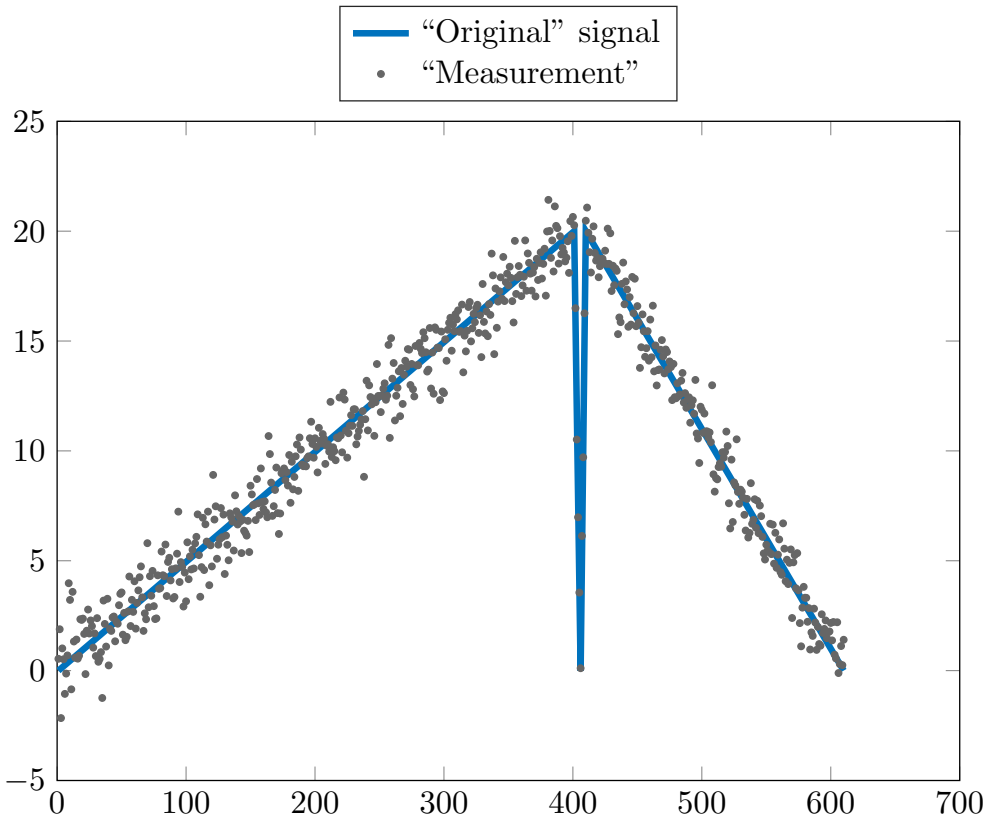


FIGURE E.1. Signal with "sloop" shape, with accompanying "measurement noise"

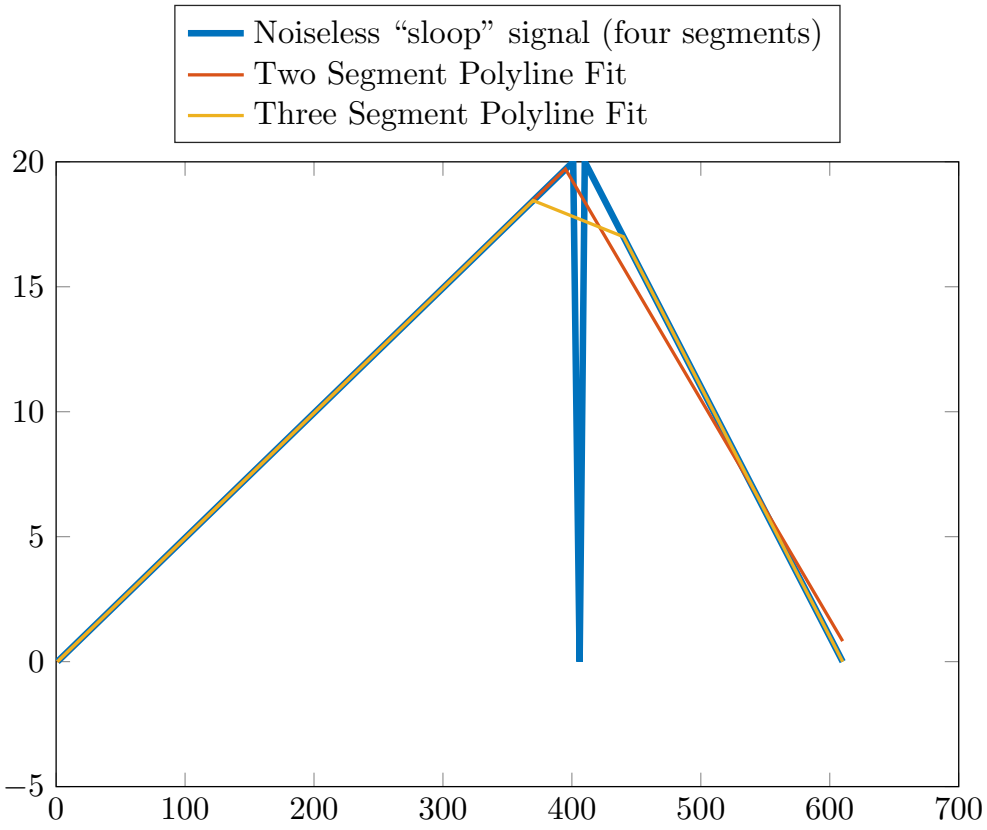


FIGURE E.2. Noiseless signal with “sloop” shape and minimum-MSE polyline approximations by segment count

TABLE E.1. Residual error and bicriterion-optimality, by segment count, for the noiseless “sloop” polyline approximations of Figure E.2

Segments	Solution	Cost	Optimal When
2	(1, -.0062) – (395, 19.71) – (610, 0.82)	1153.8+ ζ	$576.9 < \zeta < 17162$
3	(1, -.0004) – (370, 18.45) – (440, 17.00) – (610, -.0007)	971.3+ 2 ζ	Never
4	(1, 0) – (401, 20) – (406, 0) – (410, 20) – (610, 0)	0.0+ 3 ζ	$\zeta < 576.9$

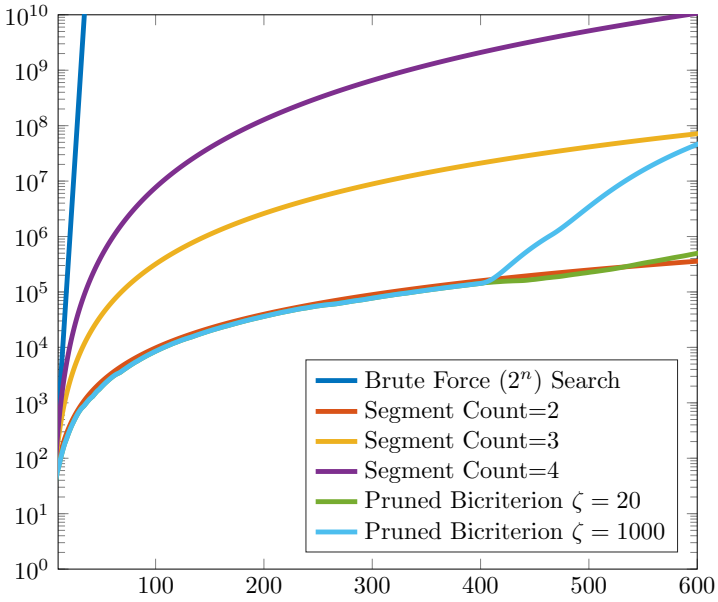


FIGURE E.3. Comparison of computational complexity of residual minimization using fixed segment count and bicriterion regularization with and without pruning, signal is the noiseless "sloop" shown in Figure E.1

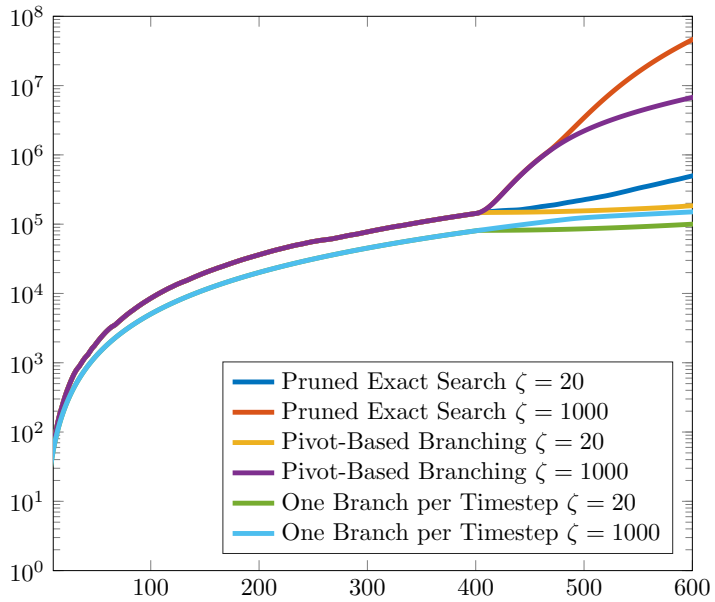


FIGURE E.4. Comparison of computational complexity of bicriterion regularization via pruned dynamic programming, with and without truncation heuristics, applied to the noiseless "sloop" shown in Figure E.1

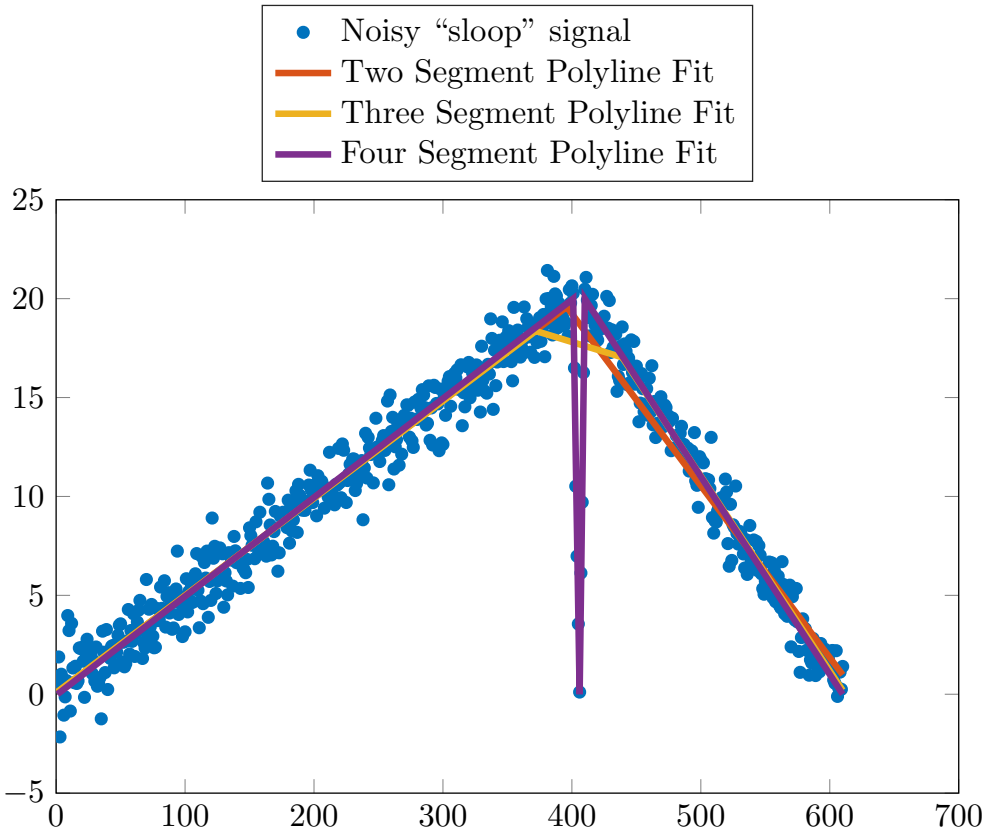


FIGURE E.5. Minimum-MSE polyline approximations for “sloop with measurement noise” signal

TABLE E.2. Residual error and bicriterion-optimality, by segment count, for the “sloop with measurement noise” polyline approximations of Figure E.5

Segments	Solution	Cost	Optimal When
2	(1, .1071) – (396, 19.5805) – (610, 0.9810)	1757.2+ ζ	$579.4 < \zeta < 16636$
3	(1, .1235) – (372, 18.37) – (439, 17.03) – (610, .1964)	1578.6+ 2 ζ	Never
4	(1, .1058) – (401, 19.83) – (406, -.0575) – (410, 19.96) – (610, 0.1636)	598.3+ 3 ζ	$6.9 < \zeta < 579.4$
5	(1, .8426) – (34, 1.608) – (401, 19.90) – (406, -.0751) – (410, 19.96) – (610, 0.1635)	591.5+ 4 ζ	$\zeta < 6.9$

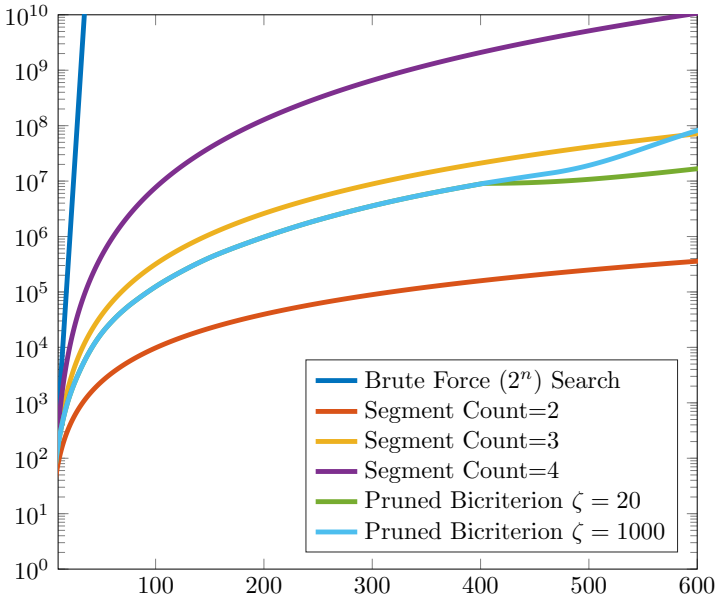


FIGURE E.6. Comparison of computational complexity of residual minimization using fixed segment count and bicriterion regularization with and without pruning, signal is the "sloop with measurement noise" shown in Figure E.1

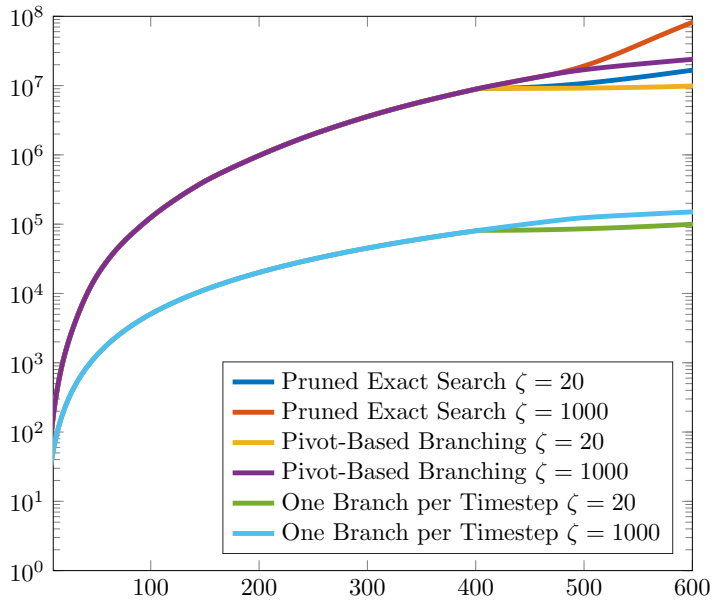


FIGURE E.7. Comparison of computational complexity of bicriterion regularization via pruned dynamic programming, with and without truncation heuristics, applied to the "sloop with measurement noise" shown in Figure E.1

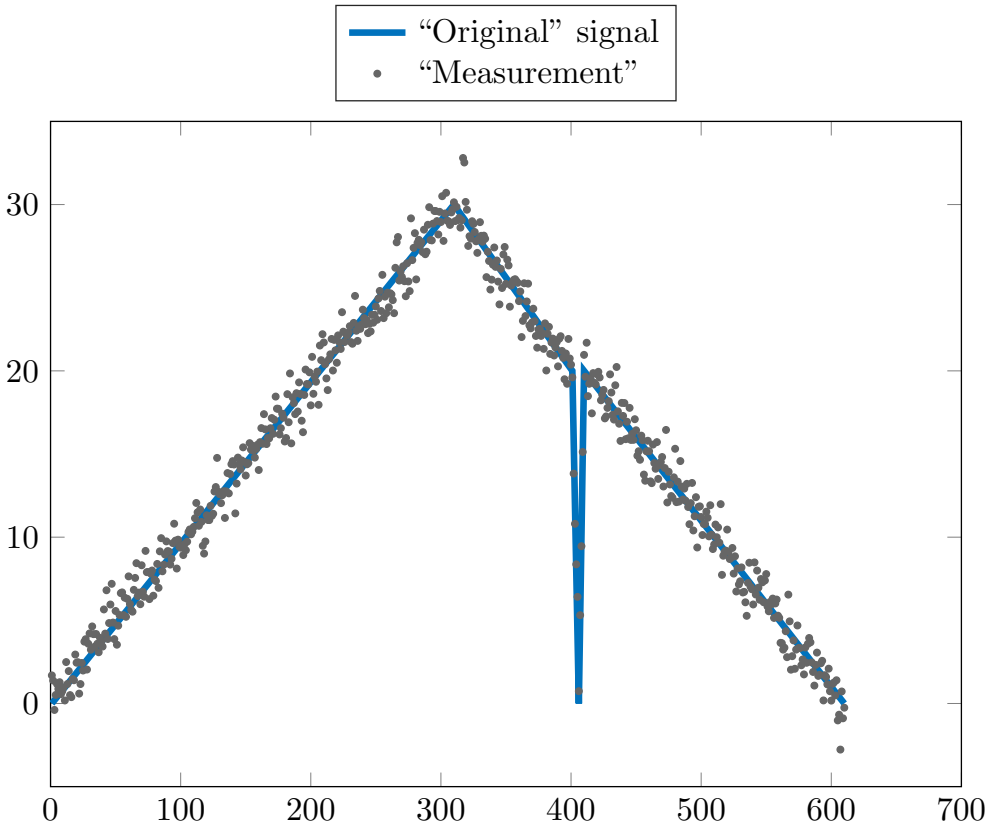


FIGURE E.8. Signal with "gaff cutter" shape, with accompanying "measurement noise"

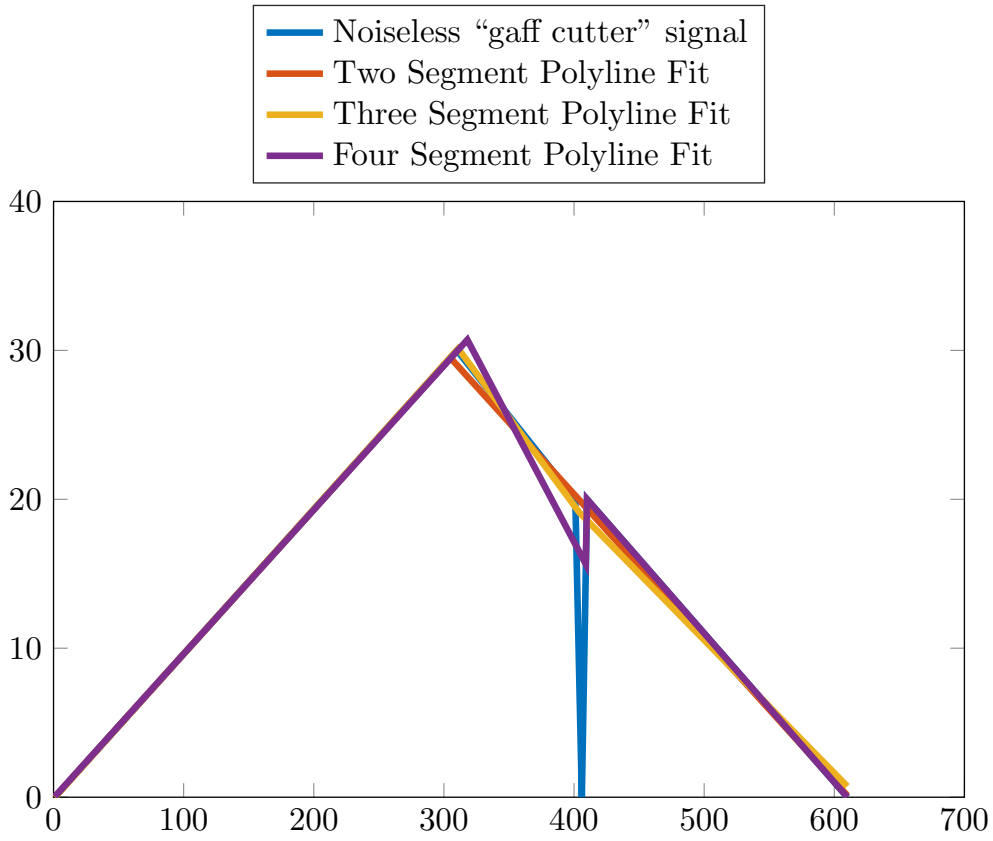


FIGURE E.9. Noiseless signal with “gaff cutter” shape and minimum-MSE polyline approximations for several segment counts

TABLE E.3. Residual error and bicriterion-optimality, by segment count, for the noiseless “gaff cutter” polyline approximations of Figure E.9

Segments	Solution	Cost	Optimal When
2	(1, .0082) – (305, 29.4980) – (610, .1085)	1241.9+ ζ	$414 < \zeta < 44202$
3	(1, -.0084) – (311, 30.11) – (404, 19.12) – (610, .7222)	1162.1+ 2 ζ	Never
4	(1, .0329) – (318, 30.71) – (409, 15.65) – (410, 20) – (610, 0)	862.8+ 3 ζ	Never
5	(1, 0) – (310, 30) – (401, 20) – (406, 0) – (410, 20) – (610, 0)	0.0+ 4 ζ	$\zeta < 414$

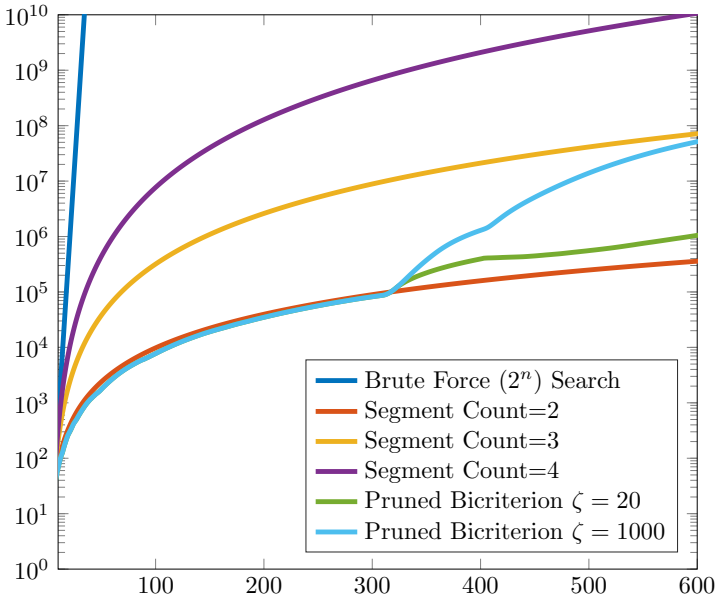


FIGURE E.10. Comparison of computational complexity of residual minimization using fixed segment count and bicriterion regularization with and without pruning, signal is the noiseless "gaff cutter" shown in E.8

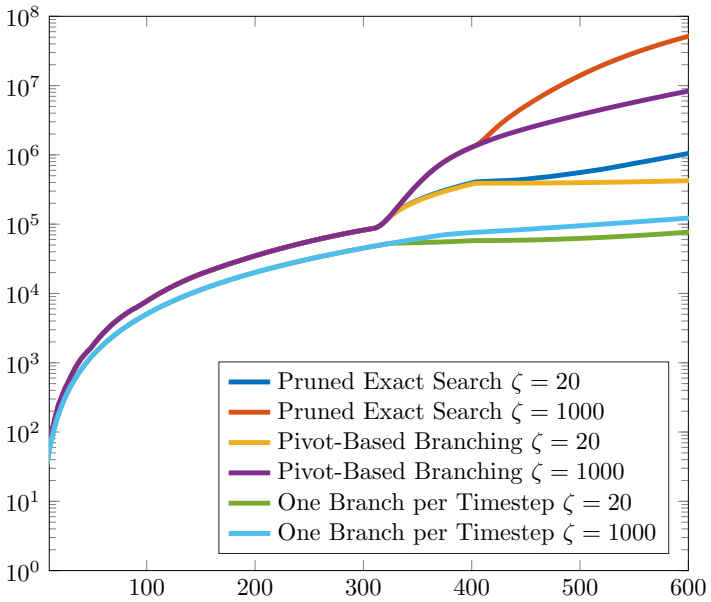


FIGURE E.11. Comparison of computational complexity of bicriterion regularization via pruned dynamic programming, with and without truncation heuristics, applied to the noiseless "gaff cutter" shown in Figure E.8

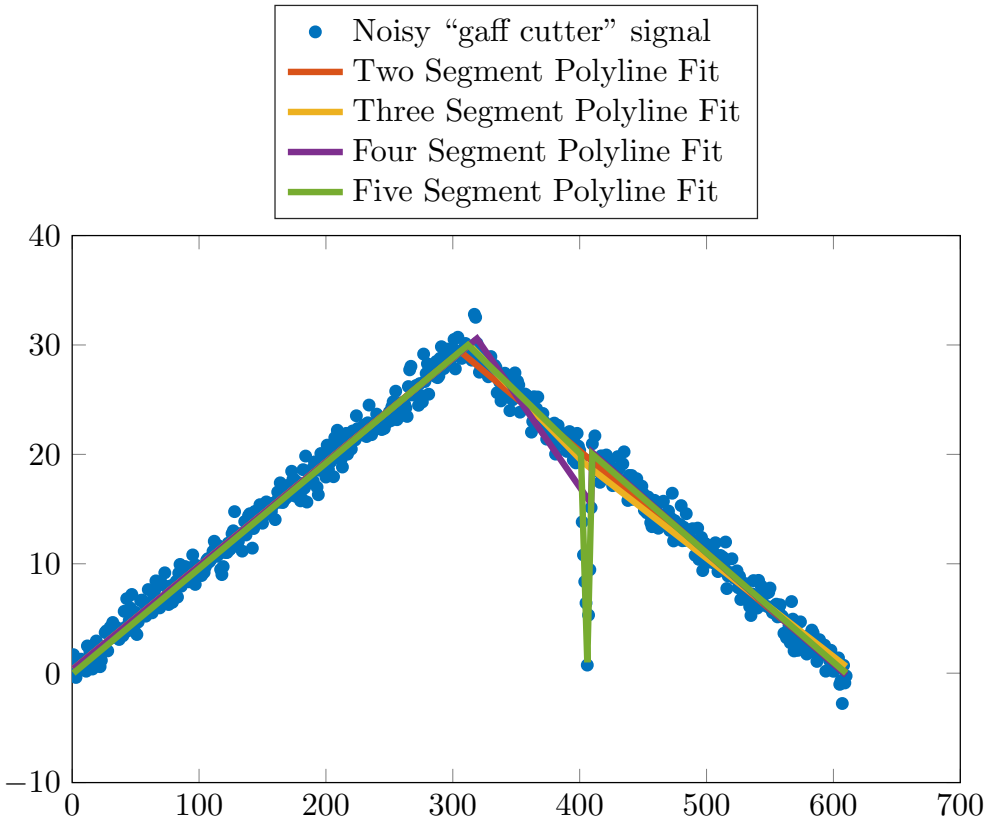


FIGURE E.12. Optimal polyline approximations for “gaff cutter” signal with added noise and with fixed segment count

TABLE E.4. Residual error and optimality, by segment count, for the noisy “gaff cutter” signal of Figure E.12

Segments	Solution	Cost	Optimal When
2	(1, .3783) – (306, 29.3935) – (610, .0530)	1752.0+ ζ	$392.8 < \zeta < 43420$
3	(1, .4028) – (313, 30.01) – (403, 19.24) – (610, .6193)	1681.0+ 2 ζ	Never
4	(1, .4025) – (319, 30.58) – (409, 15.50) – (410, 20.16) – (610, -.1739)	1330.9+ 3 ζ	Never
5	(1, 0) – (312, 29.94) – (401, 19.70) – (406, .6872) – (410, 20.15) – (610, -.1675)	573.6+ 4 ζ	$\zeta < 392.8$

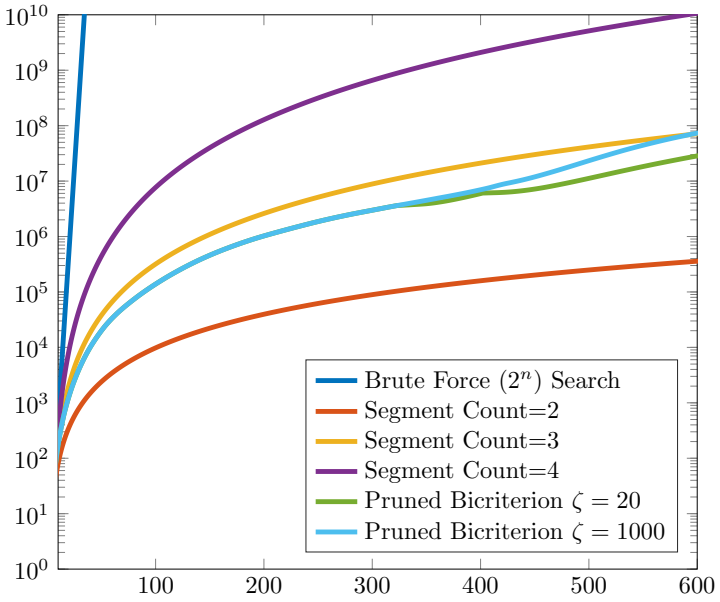


FIGURE E.13. Comparison of computational complexity of residual minimization using fixed segment count and bicriterion regularization with and without pruning, signal is the "gaff cutter with measurement noise" shown in E.8

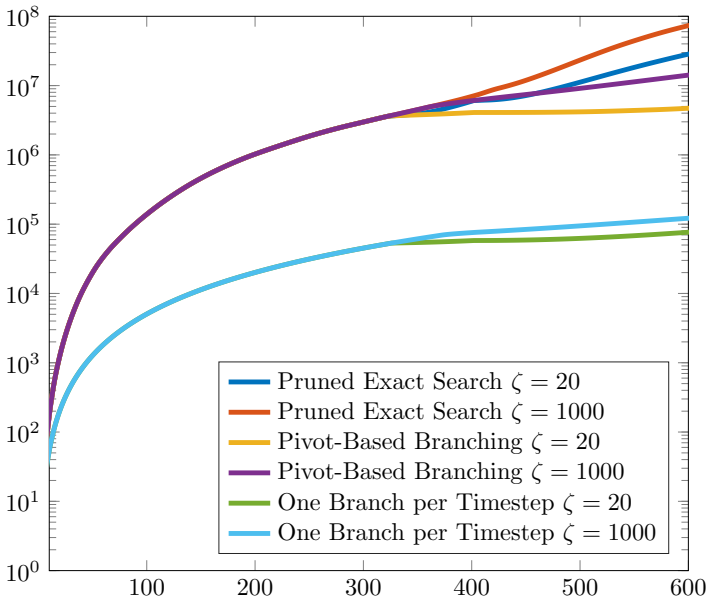


FIGURE E.14. Comparison of computational complexity of bicriterion regularization via pruned dynamic programming, with and without truncation heuristics, applied to the "gaff cutter with measurement noise" shown in Figure E.8

APPENDIX F

Changelog

The following errata in the submitted dissertation have been corrected in this version:

- In section 3.2 as well as Figure 6 and Figure 7, I incorrectly attributed the S&P 500 dataset to Little and Jones (2010). The correct source for the dataset and the ℓ^1 trend filtering analysis is Kim et al. (2009), as stated in paragraph 1.3.3.2.
- In section 5.2, the word "with" was an artifact of an earlier sentence structure, and has been corrected to "when".

Bibliography

Massimo Antonelli, Mitchell Levy, Peter J. D. Andrews, Jean Chastre, Leonard D. Hudson, Constantine Manthous, G. Umberto Meduri, Rui P. Moreno, Christian Putensen, Thomas Stewart, and et al. Hemodynamic monitoring in shock and implications for management. *Intensive Care Medicine*, 33(4):575–590, Feb 2007. doi: 10.1007/s00134-007-0531-4.

S. Arlati, E. Storti, V. Pradella, L. Bucci, A. Vitolo, and M. Pulici. Decreased fluid volume to reduce organ damage: A new approach to burn shock resuscitation? a preliminary study. *Resuscitation*, 72(3):371–378, Mar 2007. doi: 10.1016/j.resuscitation.2006.07.010.

Zsolt Balogh. Supranormal trauma resuscitation causes more cases of abdominal compartment syndrome. *Archives of Surgery*, 138(6):637, Jun 2003. doi: 10.1001/archsurg.138.6.637.

Zsolt Balogh, Bruce A McKinley, John B Holcomb, Charles C Miller, Christine S Cocanour, Rosemary A Kozar, Alicia Valdivia, Drue N Ware, and Frederick A Moore. Both primary and secondary abdominal compartment syndrome can be predicted early and are harbingers of multiple organ failure. *Journal of Trauma-Injury, Infection, and Critical Care*, 54(5):848–861, 2003. doi: 10.1097/01.TA.0000070166.29649.F3.

Cynthia Barnhart, Ellis L Johnson, George L Nemhauser, Martin WP Savelsbergh, and Pamela H Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations research*, 46(3):316–329, 1998. doi: 10.1287/opre.46.3.316.

Richard Bellman. On the approximation of curves by line segments using dynamic programming. *Communications of the ACM*, 4(6):284, June 1961. doi: 10.1145/366573.366611.

Richard Bellman and Robert Roth. Curve fitting by segmented straight lines. *Journal of the American Statistical Association*, 64(327):1079–1084, 1969. doi: 10.1080/01621459.1969.10501038.

Pietro Belotti, Jon Lee, Leo Liberti, François Margot, and Andreas Wächter. Branching and bounds tightening techniques for non-convex minlp. *Optimization Methods and Software*, 24 (4-5):597–634, Oct 2009. doi: 10.1080/10556780903087124.

Dimitris Bertsimas and Romy Shioda. Classification and regression via integer optimization. *Operations Research*, 55(2):252–271, 2007. doi: 10.1287/opre.1060.0360.

S. Bibian, G.A. Dumont, M. Huzmezan, and C.R. Ries. Quantifying uncertainty bounds in anesthetic PKPD models. In *Engineering in Medicine and Biology Society, 2004. IEMBS '04. 26th Annual International Conference of the IEEE*, volume 1, pages 786–789. IEEE, Sept 2004. ISBN http://id.crossref.org/isbn/0-7803-8439-3. doi: 10.1109/iembs.2004.1403276.

Stphane Bibian, Craig R. Ries, Mihai Huzmezan, and Guy Dumont. Introduction to automated drug delivery in clinical anesthesia. *European Journal of Control*, 11(6):535–557, Jan 2005. doi: 10.3166/ejc.11.535-557.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, March 2004. ISBN 9780521833783.

Leopoldo C. Cancio. Initial assessment and fluid resuscitation of burn patients. *Surgical Clinics of North America*, 94(4):741–754, Aug 2014. doi: 10.1016/j.suc.2014.05.003.

Brian C. Carter, Michael Vershinin, and Steven P. Gross. A comparison of step-detection methods: How well can you do? *Biophysical Journal*, 94(1):306–319, Jan 2008. doi: 10.1529/biophysj.107.110601.

R. Chartrand. Exact reconstruction of sparse signals via nonconvex minimization. *Signal Processing Letters, IEEE*, 14(10):707–710, Oct 2007. doi: 10.1109/LSP.2007.898300.

Cécile Cordier, Hugues Marchand, Richard Laundy, and Laurence A. Wolsey. bc-opt: a branch-and-cut code for mixed integer programs. *Mathematical Programming*, 86(2):335–353, Nov 1999. doi: 10.1007/s101070050092.

David L. Donoho. For most large underdetermined systems of linear equations the minimal ℓ_1-norm solution is also the sparsest solution. *Comm. Pure Appl. Math.*, 59(6):797–829, 2006. doi: 10.1002/cpa.20132.

Brian J. Eastridge, Robert L. Mabry, Peter Seguin, Joyce Cantrell, Terrill Tops, Paul Uribe, Olga Mallett, Tamara Zubko, Lynne Oetjen-Gerdes, Todd E. Rasmussen, and et al. Death on the battlefield (2001-2011). *Journal of Trauma and Acute Care Surgery*, 73:S431–S437, 2012. doi: 10.1097/ta.0b013e3182755dcc.

Samir M. Fakhry, Arthur L. Trask, Maureen A. Waller, and Dorraine D. Watts. Management of brain-injured patients by an evidence-based medicine protocol improves outcomes and decreases hospital charges. *The Journal of Trauma: Injury, Infection, and Critical Care*, 56(3): 492–500, 2004. doi: 10.1097/01.ta.0000115650.07193.66.

Robert Fano. A heuristic discussion of probabilistic decoding. *Information Theory, IEEE Transactions on*, 9(2):64–74, 1963.

Lucian Fodor, Adriana Fodor, Ytzhack Ramon, Oren Shoshani, Yaron Rissin, and Yehuda Ullmann. Controversies in fluid resuscitation for burn management: Literature review and our experience. *Injury*, 37(5):374–379, May 2006. doi: 10.1016/j.injury.2005.06.037.

D. Gagnon. The Frank-Starling mechanism and thermal stress: Fundamentals applied! *The Journal of Physiology*, 587(17):4147–4148, Aug 2009. doi: 10.1113/jphysiol.2009.176925.

Behnood Gholami, James M. Bailey, Wassim M. Haddad, and Allen R. Tannenbaum. Clinical decision support and closed-loop control for cardiopulmonary management and intensive care unit sedation using expert systems. *IEEE Trans Control Syst Technol*, 20(5):1343–1350, Mar 2012. doi: 10.1109/TCST.2011.2162412.

Somes C Guha, Michael P Kinsky, Brian Button, David N Herndon, Lillian D Traber, Daniel L Traber, and George C Kramer. Burn resuscitation: crystalloid versus colloid versus hypertonic saline hyperoncotic colloid in sheep. *Critical care medicine*, 24(11):1849–1857, 1996.

Scott B. Guthery. Partition regression. *Journal of the American Statistical Association*, 69(348): 945–947, Dec 1974. doi: 10.1080/01621459.1974.10480233.

Wassim M. Haddad and James M. Bailey. Closed-loop control for intensive care unit sedation. *Best Pract Res Clin Anaesthesiol*, 23(1):95–114, Mar 2009. doi: 10.1016/j.bpa.2008.07.007.

Mark A. Hamilton, Maurizio Cecconi, and Andrew Rhodes. A systematic review and meta-analysis on the use of preemptive hemodynamic intervention to improve postoperative outcomes in moderate and high-risk surgical patients. *Anesthesia & Analgesia*, 112(6):1392–1402, 2011. doi: 10.1213/ane.0b013e3181eeaae5.

Jeremy J. Hammond, Walter M. Kirkendall, and Richard V. Calfee. Hypertensive crisis managed by computer controlled infusion of sodium nitroprusside: A model for the closed loop administration of short acting vasoactive agents. *Computers and Biomedical Research*, 12(2):97–108, Apr 1979. doi: 10.1016/0010-4809(79)90008-9.

Thomas M Hemmerling. Automated anesthesia. *Current Opinion in Anaesthesiology*, 22(6): 757–763, 2009. doi: 10.1097/aco.0b013e328332c9b4.

C. Holm, M. Mayr, J. Tegeler, F. Hrbrand, G. Henckel von Donnersmarck, W. Mhlbauer, and U.J. Pfeiffer. A clinical randomized study on the effects of invasive monitoring on burn shock resuscitation. *Burns*, 30(8):798–807, Dec 2004. doi: 10.1016/j.burns.2004.06.016.

R G Holzheimer and J A Mannick, editors. *Surgical Treatment: Evidence-Based and Problem-Oriented*. Zuckschwerdt, Munich, 2001.

Stephen L. Hoskins, Geir Ivar Elgjo, Jialung Lu, Hao Ying, James J. Grady, David N. Herndon, and George C. Kramer. Closed-loop resuscitation of burn shock. *Journal of Burn Care &*

Research, 27(3):377–385, 2006. doi: 10.1097/01.bcr.0000216512.30415.78.

P. Julian, M. Jordan, and A. Desages. Canonical piecewise-linear approximation of smooth functions. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 45(5):567–571, May 1998. doi: 10.1109/81.668868.

Daniel Kahneman. *Thinking, Fast and Slow*. Farrar, Straus and Giroux, 2011. ISBN 0374275637.

Jacob W. J. Kerssemakers, E. Laura Munteanu, Liedewij Laan, Tim L. Noetzel, Marcel E. Janson, and Marileen Dogterom. Assembly dynamics of microtubules at molecular resolution. *Nature*, 442(7103):709–712, Aug 2006. doi: 10.1038/nature04928.

R. Killick, P. Fearnhead, and I. A. Eckley. Optimal detection of changepoints with a linear computational cost. *Journal of the American Statistical Association*, 107(500):1590–1598, Dec 2012. doi: 10.1080/01621459.2012.737745.

Seung-Jean Kim, Kwangmoo Koh, Stephen Boyd, and Dimitry Gorinevsky. ℓ_1 trend filtering. *SIAM Review*, 51(2):339–360, May 2009. doi: 10.1137/070690274.

H. Kobayashi. Simultaneous adaptive estimation and decision algorithm for carrier modulated data transmission systems. *IEEE Transactions on Communications*, 19(3):268–280, Jun 1971. doi: 10.1109/tcom.1971.1090639.

G C Kramer, T Lund, and O Beckum. Pathophysiology of burn shock and burn edema. In David N. Herndon, editor, *Total Burn Care*, chapter 8, pages 93–106. Saunders Elsevier, 3rd edition, 2007a. ISBN 1416032746,9781416032748.

George Kramer, Steve Hoskins, Nick Copper, Jiin-Yu Chen, Michelle Hazel, and Charles Mitchell. Emerging advances in burn resuscitation. *The Journal of Trauma: Injury, Infection, and Critical Care*, 62(Supplement):S71–S72, 2007b. doi: 10.1097/ta.0b013e318065aedf.

George C. Kramer, Michael P. Kinsky, Donald S. Prough, Jose Salinas, Jill L. Sondeen, Michelle L. Hazel-Scerbo, and Charles E. Mitchell. Closed-loop control of fluid therapy for treatment of hypovolemia. *The Journal of Trauma: Injury, Infection, and Critical Care*, 64 (Supplement):S333–S341, 2008. doi: 10.1097/ta.0b013e31816bf517.

James Stephen Krinsley. Effect of an intensive glucose management protocol on the mortality of critically ill adult patients. In *Mayo Clinic Proceedings*, volume 79, pages 992–1000. Elsevier, 2004.

Mitchell M. Levy, R. Phillip Dellinger, Sean R. Townsend, Walter T. Linde-Zwirble, John C. Marshall, Julian Bion, Christa Schorr, Antonio Artigas, Graham Ramsay, Richard Beale, Margaret M. Parker, Herwig Gerlach, Konrad Reinhart, Eliezer Silva, Maurene Harvey, Susan Regan, and Derek C. Angus. The surviving sepsis campaign: results of an international guideline-based performance improvement program targeting severe sepsis. *Intensive Care Medicine*, 36(2):222–231, Jan 2010. doi: 10.1007/s00134-009-1738-3.

Sven Leyffer, Annick Sartenaer, and Emilie Wanufelle. Branch-and-refine for mixed-integer nonconvex global optimization. Technical Report Preprint ANL/MCS-P1547-0908, Mathematics and Computer Science Division, Argonne National Laboratory, 2008.

M. A. Little and Nick S. Jones. Sparse Bayesian step-filtering for high-throughput analysis of molecular machine dynamics. In *Proceedings of the 2010 IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, pages 4162–4165, 2010. doi: 10.1109/ICASSP.2010.5495722.

Tamas Luspay and Karolos M. Grigoriadis. Design and validation of an extended Kalman filter for estimating hemodynamic variables. *2014 American Control Conference*, Jun 2014. doi: 10.1109/acc.2014.6859101.

Greg S. Martin, David M. Mannino, Stephanie Eaton, and Marc Moss. The epidemiology of sepsis in the united states from 1979 through 2000. *N Engl J Med*, 348(16):1546–1554, Apr 2003. doi: 10.1056/nejmoa022139.

Christopher L Meador. Closed loop fluid delivery system. Technical report, Arcos, Inc., Feb 2014.

Andrew Conway Morris, Alasdair W. Hay, David G. Swann, Kirsty Everingham, Corrienne McCulloch, Jane McNulty, Odette Brooks, Ian F. Laurenson, Brian Cook, and Timothy S. Walsh. Reducing ventilator-associated pneumonia in intensive care: Impact of implementing a care bundle. *Critical Care Medicine*, 39(10):2218–2224, 2011. doi: 10.1097/ccm.0b013e3182227d52.

Craig D. Newgard, Robert H. Schmicker, Jerris R. Hedges, John P. Trickett, Daniel P. Davis, Eileen M. Bulger, Tom P. Aufderheide, Joseph P. Minei, J. Steven Hata, K. Dean Gubler, and et al. Emergency medical services intervals and survival in trauma: Assessment of the "golden hour" in a north american prospective cohort. *Annals of Emergency Medicine*, 55(3): 235–246.e4, Mar 2010. doi: 10.1016/j.annemergmed.2009.07.024.

Warwick D. Ngan Kee, Anna Lee, Kim S. Khaw, Floria F. Ng, Manoj K. Karmakar, and Tony Gin. A randomized double-blinded comparison of phenylephrine and ephedrine infusion combinations to maintain blood pressure during spinal anesthesia for cesarean delivery: The effects on fetal acid-base status and hemodynamic control. *Anesthesia & Analgesia*, 107(4):1295–1302, Oct 2008. doi: 10.1213/ane.0b013e31818065bc.

Jun Oda, Katsuyuki Yamashita, Takuya Inoue, Nobuyuki Harunari, Yasumasa Ode, Kazuharu Mega, Yoshiki Aoki, Mitsuhiro Noborio, and Masashi Ueyama. Resuscitation fluid volume and abdominal compartment syndrome in patients with major burns. *Burns*, 32(2):151154, Mar 2006. doi: 10.1016/j.burns.2005.08.011.

ONR BAA 11-012. *BROAD AGENCY ANNOUNCEMENT (BAA) Autonomous Critical Care System (ACCS)*. Department of the Navy, 2011.

ONR BAA 12-004 CONOPs. *Autonomous Aerial Cargo/Utility System (AACUS) Concept of Operations (CONOPs)*. Department of the Navy, 2012.

Ronny M. Otero. Early goal-directed therapy in severe sepsis and septic shock revisited. *CHEST Journal*, 130(5):1579, Nov 2006. doi: 10.1378/chest.130.5.1579.

Azriel Perel. Bench-to-bedside review: The initial hemodynamic resuscitation of the septic patient according to surviving sepsis campaign guidelines does one size fit all? *Crit Care*, 12 (5):223, 2008. doi: 10.1186/cc6979.

Tam N. MD Pham, Leopoldo C. MD Cancio, and Nicole S. MD Gibran. American burn association practice guidelines burn shock resuscitation. *Journal of Burn Care & Research*, 29(1): 257–266, January/February 2008.

S. Preisman. Predicting fluid responsiveness in patients undergoing cardiac surgery: functional haemodynamic parameters including the respiratory systolic variation test and static preload indicators. *British Journal of Anaesthesia*, 95(6):746–755, Oct 2005. doi: 10.1093/bja/aei262.

Abraham D Rafie, Paul A Rath, Michael W Michell, Robert A Kirschner, Donald J Deyo, Donald S Prough, James J Grady, and George C Kramer. Hypotensive resuscitation of multiple hemorrhages using crystalloid and colloids. *Shock*, 22(3):262–269, 2004. doi: 10.1097/01.shk.0000135255.59817.8c.

Carlos Ramirez, Vladik Kreinovich, and Miguel Argaez. Why ℓ^1 is a good approximation to ℓ^0: A geometric explanation. *Journal of Uncertain Systems*, 7(3):203–207, 2013.

R.R. Rao, B. Aufderheide, and B.W. Bequette. Multiple model predictive control of hemodynamic variables: an experimental study. In *American Control Conference, Proceedings of the 1999*, volume 2, pages 1253–1257. IEEE, Jun 1999. ISBN http://id.crossref.org/isbn/0-7803-4990-3. doi: 10.1109/acc.1999.783568.

R.R. Rao, B. Aufderheide, and B.W. Bequette. Experimental studies on multiple-model predictive control for automated regulation of hemodynamic variables. *IEEE Trans. Biomed. Eng.*, 50(3):277–288, Mar 2003. doi: 10.1109/tbme.2003.808813.

Guillem Rigaill. Pruned dynamic programming for optimal multiple change-point detection. arXiv preprint arXiv:1004.0887, 2010.

Joseph Rinehart, Brenton Alexander, Yannick Manach, Christoph Hofer, Benoit Tavernier, Zeev N Kain, and Maxime Cannesson. Evaluation of a novel closed-loop fluid-administration system based on dynamic predictors of fluid responsiveness: an in silico simulation study. *Crit*

Care, 15(6):R278, 2011. doi: 10.1186/cc10562.

Joseph Rinehart, Ngai Liu, Brenton Alexander, and Maxime Cannesson. Closed-loop systems in anesthesia. *Anesthesia & Analgesia*, 114(1):130–143, Jan 2012. doi: 10.1213/ane.0b013e318230e9e0.

Jacob Roll, Alberto Bemporad, and Lennart Ljung. Identification of piecewise affine systems via mixed-integer programming. *Automatica*, 40(1):37–50, Jan 2004. doi: 10.1016/j.automatica.2003.08.006.

Hong S Ryoo and Nikolaos V Sahinidis. A branch-and-reduce approach to global optimization. *Journal of Global Optimization*, 8(2):107–138, 1996.

Jeffrey R. Saffle. The phenomenon of "fluid creep" in acute burn resuscitation. *Journal of Burn Care & Research*, 28(3):382–395, 2007. doi: 10.1097/bcr.0b013e318053d3a1.

B H Saggi, R Ivatury, and Sugerman HJ. *Abdominal compartment syndrome*, chapter Part XVI. Surgical critical care issues. In Holzheimer and Mannick (2001), 2001.

Nikolaos V Sahinidis. Baron: A general purpose global optimization software package. *Journal of global optimization*, 8(2):201–205, 1996.

Jose Salinas, Guy Drew, James Gallagher, Leopoldo C. Cancio, Steven E. Wolf, Charles E. Wade, John B. Holcomb, David N. Herndon, and George C. Kramer. Closed-loop and decision-assist resuscitation of burn patients. *The Journal of Trauma: Injury, Infection, and Critical Care*, 64(Supplement):S321–S332, 2008. doi: 10.1097/ta.0b013e31816bf4f7.

Jose Salinas, Kevin K Chung, Elizabeth A Mann, Leopoldo C Cancio, George C Kramer, Maria L Serio-Melvin, Evan M Renz, Charles E Wade, and Steven E Wolf. Computerized decision support system improves fluid resuscitation following severe burns: An original study*. *Critical care medicine*, 39(9):2031–2038, 2011. doi: 10.1097/CCM.0b013e31821cb790.

A P Sanford and D N Herndon. *Current therapy of burns*, chapter Part XV. Surgical Infections. In Holzheimer and Mannick (2001), 2001.

Philip C. Spinella and John B. Holcomb. Resuscitation and transfusion principles for traumatic hemorrhagic shock. *Blood Reviews*, 23(6):231–240, Nov 2009. doi: 10.1016/j.blre.2009.07.003.

Michel MRF Struys, Tom De Smet, Linda FM Versichelen, Stijn Van de Velde, Rudy Van den Broecke, and Eric P Mortier. Comparison of closed-loop controlled administration of propofol using bispectral index as the controlled variable versus "standard practice" controlled administration. *Anesthesiology*, 95(1):6–17, 2001.

Michael Sugrue. Abdominal compartment syndrome. *Current opinion in critical care*, 11(4):333–338, 2005.

Llewellyn Hilleth Thomas. Elliptic problems in linear difference equations over a network. *Watson Sci. Comput. Lab. Rept., Columbia University, New York*, 1949.

Alejandro Toriello and Juan Pablo Vielma. Fitting piecewise linear continuous functions. *European Journal of Operational Research*, 219(1):86–95, May 2012. doi: 10.1016/j.ejor.2011.12.030.

Betty J Tsuei and Paul A Kearney. Hypothermia in the trauma patient. *Injury*, 35(1):7–15, Jan 2004. doi: 10.1016/s0020-1383(03)00309-7.

Sumreen U. Vaid, Alia Shah, Michael W. Michell, Abraham D. Rafie, Donald J. Deyo, Donald S. Prough, and George C. Kramer. Normotensive and hypotensive closed-loop resuscitation using 3.0% NaCl to treat multiple hemorrhages in sheep. *Critical Care Medicine*, 34(4):1185–1192, 2006. doi: 10.1097/01.ccm.0000207341.78696.3a.

Melanie van der Heijden, Joanne Verheij, Geerten P. van Nieuw Amerongen, and A B. Johan Groeneveld. Crystalloid or colloid fluid loading and pulmonary permeability, edema, and injury in septic and nonseptic critically ill patients with hypovolemia*. *Critical Care Medicine*, 37(4):1275–1281, 2009. doi: 10.1097/ccm.0b013e31819cedfd.

A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inform. Theory*, 13(2):260–269, Apr 1967. doi: 10.1109/tit.1967.1054010.

Kelly A Wood and Derek C Angus. Pharmacoeconomic implications of new therapies in sepsis. *PharmacoEconomics*, 22(14):895–906, 2004. doi: 10.2165/00019053-200422140-00001.

Li Xu, Cewu Lu, Yi Xu, and Jiaya Jia. Image smoothing via L_0 gradient minimization. *ACM Transactions on Graphics*, 30(6):1, Dec 2011. doi: 10.1145/2070781.2024208.

Yi-Ching Yao. Estimation of a noisy discrete-time step function: Bayes and empirical Bayes approaches. *Ann. Statist.*, 12(4):1434–1447, Dec 1984. doi: 10.1214/aos/1176346802.

H. Ying, M. McEachern, D.W. Eddleman, and L.C. Sheppard. Fuzzy control of mean arterial pressure in postsurgical patients with sodium nitroprusside infusion. *IEEE Trans. Biomed. Eng.*, 39(10):10601070, 1992. doi: 10.1109/10.161338.

H. Ying, C.A. Bonnerup, R.A. Kirschner, D.J. Deyo, M.W. Michell, and G.C. Kramer. Closed-loop fuzzy control of resuscitation of hemorrhagic shock in sheep. In *Engineering in Medicine and Biology, 2002. 24th Annual Conference and the Annual Fall Meeting of the Biomedical Engineering Society EMBS/BMES Conference, 2002. Proceedings of the Second Joint*, volume 2, pages 1575–1576, 2002. ISBN http://id.crossref.org/isbn/0-7803-7612-9. doi: 10.1109/iembs.2002.1106545.

C. Yu, R.J. Roy, H. Kaufman, and B.W. Bequette. Multiple-model adaptive predictive control of mean arterial pressure and cardiac output. *IEEE Trans. Biomed. Eng.*, 39(8):765–778, 1992. doi: 10.1109/10.148385.

T Zikov and S Bibian. Neurowave closed-loop sedation technology. In *Military Health 2014*, Ft. Lauderdale, FL, Aug 2014. NeuroWave Systems, Inc., Smart Monitoring.