A Conceptual Framework for Consistency, Conditioning and Stability Issues in Signal Processing

J.R. Bunch, R.C. Le Borne*, I.K. Proudler

EDICS: 2-ALGO

Abstract

The techniques employed for analyzing algorithms in numerical linear algebra have evolved significantly since the 1940's. Significant with this evolution is the partitioning of the terminology into categories in which analyses involving infinite precision effects are distinguished from analyses involving finite precision effects. Although the structure of algorithms in signal processing prevents the direct application of typical analysis techniques employed in numerical linear algebra, much can be gained in signal processing from an assimilation of the terminology found there. This paper addresses the need for a conceptual framework for discussing the computed solution from an algorithm by focusing on the distinction between a perturbation analysis of a problem or a method of solution and the stability analysis of an algorithm. A consistent approach to defining these concepts facilitates the task of assessing the numerical quality of a computed solution. This paper discusses numerical analysis techniques for signal processing algorithms and suggests terminology supportive for a centralized framework for distinguishing between errors propagated by the nature of the problem and errors propagated through the use of finite precision arithmetic. By this means the numerical stability analysis of a signal processing algorithm can be simplified and the meaning of such an analysis made unequivocal.

Keywords

Adaptive filters, analysis (convergence, tracking, stability, numerical)

The authors appear alphabetically.

Partial support for J.R. Bunch and R.C. Le Borne was provided by NSF grant MIP-9625482.

J.R. Bunch: University of California, San Diego,Department of Mathematics, La Jolla, CA 92093-0112, USA. phone: +001 619 534-4889, fax: +001 619 534-5273, e-mail: jbunch@ucsd.edu

R.C. Le Borne*: Technical University Hamburg-Harburg, Department of Mathematics, D-21071 Hamburg, Germany. phone: +49 40 42878 3670, fax: +49 40 42878 2696, e-mail: leborne@tu-harburg.de

I.K. Proudler: DERA, St. Andrews Road, Malvern Worcs. WR14 3PS, United Kingdom. phone: +44 (0)1684 894228, fax: +44 (0)1684 896502, e-mail: proudler@signal.dera.gov.uk

I. Introduction

In the past, guidelines for assessing the numerical quality of an algorithm in signal processing have been quite flexible. Authors have been permitted to take a liberal view of what is meant by stability, its definition often drawn from different fields, and as a result it has not been clearly understood by the subsequent reader of the analysis. Furthermore, this is not restricted to fields pertaining to signal processing. General examples include such diverse fields as differential equations, linear multi-step methods and economic models, while fields particular to signal processing include dynamical systems, control theory and numerical linear algebra. The diversity of stability terminology includes BIBO, Lyapunov and backwards stability, to name a few. With the ultimate desire for an algorithmic analysis to establish confidence that (or the conditions in which) the computed solution is usable, a lack of clarity regarding the application of the analysis with respect to the class of stability will unnecessarily reduce its overall effectiveness. It has therefore become clear that a necessary precursor to a full and comprehensive understanding of the many important existing (and future) analyses of signal processing algorithms is the categorical use of the term "stability". Additionally, before one can achieve this objective, the relationship between an algorithm's stability and a problem's conditioning must be well-understood.

We will treat perturbations that are derived from computer arithmetic differently from all other types. The necessity for this duplication of notation is due to the fact that the ultimate behavior of a computer-based algorithm is jointly dependent on the effects of the computer as well as the effects from other perturbations such as how the problem is formulated, the method, and/or the expected inaccuracies in the data. This will be further clarified later in the text. Although the treatment of each type of perturbation is drawn from the same foundation, the terminology will be chosen to retain a distinction. With respect to the natural transformation of a problem to its formulation that ultimately becomes an algorithm, this terminology will facilitate the ability to communicate accurately where in this process the perturbation, and hence the analysis, applies. Furthermore, since valuable insight may be given from an analysis addressing a part of this transformation (from problem identification to algorithm implementation), it is important to make clear which part is under study:

- the conditioning of the problem,
- the stability of the representation of the problem (the method),

- the arithmetic reliability (arithmetic stability) of the algorithm

– inherent behavior to computer related errors in the input (independent of the organization and the choice of elementary computer operations),

– algorithmic behavior in the presence of errors caused by computer arithmetic (dependent upon the organization and the choice of elementary computer operations).

The elementary computer operations are intended to mean the normal arithmetic operations $(+, -, \times, \div)$ as well as the standard functions $(\sqrt{\cdot}, \sin, \exp, \log, \text{etc.})$. For our purposes, we will also consider an expression to be an elementary operation when the sequence of calculations is uniquely defined by the normal hierarchy of mathematical operations. For example, $a \times b + c$ will be considered to be an elementary operation whereas the expression $a \times b \times c$ will not. The two subcategories under algorithmic stability are given in order to measure the influence of imprecise arithmetic on the desired solution relative to the natural, or inherent, sensitivity of the algorithm. As will be detailed later, this inherent sensitivity can be thought of as a type of stability analysis of the method when the perturbed input is defined in a particular way.

This paper allows complete freedom regarding the tools that one may use in order to analyse an algorithm. Its purpose is to provide a clear, concise and conceptual framework from which the results of an analysis may be applied. Meeting this objective will result in a more comprehensive understanding of the value of previous analyses while facilitating the process of disseminating the full importance of any future analysis. In this paper we re-introduce general terminology connecting the conditioning of a problem to the stability of an algorithm that exists in the field of numerical analysis [1], [2]. In Section II we formally address the connection between convergence, stability, and consistency with respect to exact arithmetic. Section III involves these notions with respect to finite precision arithmetic but distinguishes its terminology. Two main analysis approaches, the forward and backward, are detailed in Sections IV and V, respectively, while the comparative analysis of multiple algorithms is addressed in Section VI. In Section VII we apply the definitions presented here to the work of Ljung and Ljung [3] in which the error propagation properties of recursive least-squares algorithms are analysed.

II. Exact Arithmetic

Suppose a problem $\mathcal{P}$ consists of determining some unknown $x \in \mathbb{R}^n$ which is given by

$$\phi(x) = y, \quad y \in \mathbb{R}^m, \tag{1}$$

i.e., ϕ is the mapping $\phi : X \longrightarrow Y$. This problem is considered well-posed in the sense of Hadamard if x exists, is unique, and the inverse mapping ϕ^{-1} is continuous. It is with the continuity of ϕ^{-1} that the term stability can be applied. If one chooses to formulate the inverse directly, we may simply write

$$\Phi(y) = x, \quad \text{where } \Phi := \phi^{-1} \tag{2}$$

is a representation of ϕ^{-1}, i.e., only the results are the same; the computational routes, and hence the conditions for continuity, may differ. As is often the case, we do not know y, but instead an approximation $y_\delta = y(1+\delta), 0 \leq \delta < 1$. Likewise, it is either impossible or undesirable to involve (1) (or (2)) directly, but rather an approximation ϕ_δ (or equivalently, Φ_δ) is substituted to give an approximation x_δ to x. Under this formulation, and assuming $\mathcal{P}$ is well-posed, the parameter δ defines the family of approximations

$$\phi_\delta(x_\delta) = y_\delta, \quad \text{or} \tag{3}$$

$$\Phi_\delta(y_\delta) = x_\delta \tag{4}$$

for which we are interested in establishing general criteria to ensure the convergence of the solution, i.e., $x_\delta \longrightarrow x$ as $\delta \longrightarrow 0$. As hinted, continuity (pointwise and uniform) is an important cornerstone for this notion of convergence. When implementational concerns such as finite precision arithmetic are ignored, and assuming some means to measure expressions $(\cdot)$, denoted by $\mathcal{M}$, we can express the direct error (residual error, resp.) $\Phi_\delta(y_\delta) - \Phi(y)$, $(\phi_\delta(x_\delta) - \phi(x)$, resp.) through,

$$\underbrace{\Phi_\delta(y_\delta) - \Phi(y)}_{\mathcal{M}(\text{direct convergence})} = \underbrace{\Phi_\delta(y_\delta) - \Phi_\delta(y)}_{\mathcal{M}(\text{stability})} + \underbrace{\Phi_\delta(y) - \Phi(y)}_{\mathcal{M}(\text{direct consistency})} \tag{5}$$

or,

$$\underbrace{\phi_\delta(x_\delta) - \phi(x)}_{\mathcal{M}(\text{res. convergence})} = \underbrace{\phi_\delta(x_\delta) - \phi_\delta(x)}_{\mathcal{M}(\text{stability})} + \underbrace{\phi_\delta(x) - \phi(x).}_{\mathcal{M}(\text{res. consistency})} \tag{6}$$

Hence, we see in (5) that when there is no perturbation in the formulation Φ of the problem, we have that stability implies convergence. When this is not the case, equations (5) and

(6) suggest that stability plus consistency implies convergence. These notions will now be formalized.

Definition 1: A set of mappings $\{\phi_\delta : X \longrightarrow Y\}$ is *equicontinuous at x* if for all $0 \leq \delta < 1$ and for all $\zeta > 0$ there exists an $\eta > 0$, independent of δ such that $\| \phi_\delta(\hat{x}) - \phi_\delta(x) \| \leq \zeta$ whenever $\| \hat{x} - x \|_X \leq \eta$.

It is easy to see, through the immediate consequence of the continuity of ϕ and its inverse, that convergence independent of δ is ensured if and only if ϕ is equicontinuous at x, i.e., $x_\delta \longrightarrow x$ as $\delta \longrightarrow 0$ if and only if $\phi(x_\delta) \longrightarrow y$. If, on the other hand, we are interested in the convergence properties of ϕ_δ, then we need the notion of consistency in addition to equicontinuity.

Definition 2: The mapping ϕ_δ is *consistent* with respect to ϕ if for some $\eta > 0$, $\phi_\delta(\hat{x}) \longrightarrow \phi(\hat{x})$ as $\delta \longrightarrow 0$ whenever $\| \hat{x} - x \| < \eta$.

Definition 3: $\Phi_\delta = \phi_\delta^{-1}$ is an approximation to the inverse mapping ϕ^{-1} if and only if ϕ_δ is consistent with respect to the mapping ϕ.

Theorem 1: For ϕ_δ consistent with respect to ϕ, a sufficient condition for convergence is the equicontinuity of Φ_δ for all $0 \leq \delta < 1$.

The proof of Theorem 1 can be found in [1, pg. 10].

Definition 4: A *method* provides in sufficient detail the intended manner (formulation) in which a problem is to be solved in exact arithmetic.

We now consider the formulation of the problem ϕ_δ or Φ_δ as the approximation method.

Definition 5: The approximation method ϕ_δ or Φ_δ is *stable* with respect to δ whenever Φ_δ is equicontinuous for $0 \leq \delta < 1$.

With the above definitions and theorem, the following important and useful result follows.

Theorem 2 (The Lax Principle) Consistency and stability imply convergence.

III. Finite Precision Arithmetic

In order to take into consideration the limitations of computer arithmetic one can again make use of the notion of asymptotic convergence: First, however, one must establish a parameter that sufficiently models the numeric structure on a computer while functioning in place of δ from Section II. One possibility is p, the digits available for arithmetic in base β on the computer. For this we have that as $p \longrightarrow \infty$, results from finite precision arithmetic will agree asymptotically to those from exact arithmetic. The alternative that will be adopted

here, however, provides this asymptotic property via the computer's representation error, ϵ (e.g., unit roundoff of the computer or the machine precision); again, $\epsilon \longrightarrow 0$ implies that results from finite precision arithmetic will agree asymptotically to those from exact arithmetic. Given this, if we now restrict a problem's formulation to the set of computer numbers, the problem $\mathcal{P}$ then is represented by the nearby and related problem $\mathcal{P}_\epsilon$ in which the exact formulation of (1) or (2) becomes

$$\Phi_\epsilon(y_\epsilon) = x_\epsilon, \quad \Phi := \phi^{-1}, \tag{7}$$

or

$$\phi_\epsilon(x_\epsilon) = y_\epsilon, \quad y \in \mathbb{R}^m. \tag{8}$$

Here Φ_ϵ (ϕ_ϵ, respectively) represents the implementation of the formulation of the problem (2) (or (1), respectively).

Definition 6: An *algorithm* describes in exact detail the finite sequence of elementary computer operations of a method intended to solve the problem.

The issue of convergence previously given under the parameter δ now describes the computability of the solution under the parameter ϵ:

Definition 7: Let Φ_ϵ be the implementation of Φ on a computer such that $\Phi_\epsilon(y_\epsilon) = x_\epsilon$ and $\Phi(y) = x$. Then x is *computable* in finite precision by Φ_ϵ if and only if $x_\epsilon \longrightarrow x$ as $\epsilon \longrightarrow 0$.

Definition 1 can be used to define equicontinuity with respect to the parameter ϵ with the obvious exchange of parameter, $\delta \leftarrow \epsilon$. With this, stability can now defined through the equicontinuity of Φ while the Lax Principle given in Theorem 1 tells us how computability is related to the notion of arithmetic stability and reliability. Again, by letting $\mathcal{M}(\cdot)$ denote a measure for expression $(\cdot)$, we have

$$\underbrace{x - x_\epsilon}_{\mathcal{M}(\text{Computability})} = \Phi(y) - \Phi_\epsilon(y_\epsilon) \tag{9}$$

$$= \underbrace{\Phi(y) - \Phi(y_\epsilon)}_{\mathcal{M}(\text{arith. stability})} + \underbrace{\Phi(y_\epsilon) - \Phi_\epsilon(y_\epsilon).}_{\mathcal{M}(\text{arith. reliability})} \tag{10}$$

The notion of arithmetic stability follows Definition 5 and the equicontinuity of Φ with respect to a neighborhood of y. Arithmetic reliability, on the other hand, applies to the degree in which the consistency of Φ_ϵ with respect to Φ at y_ϵ is tolerated. In either case the perturbation is determined by computer arithmetic. The point to underscore, nonetheless, is that the notion

of arithmetic stability does not involve the effects from a computer directly; the criteria for arithmetic stability requires that the computer representation of y be sufficiently close to y such that the definition for stability is satisfied.

IV. Forward Error Analysis

The algorithm Φ_ϵ is the computer implementation of the desired formulation for $x = \phi^{-1}(y)$. When considering the direct error, $\|x_\epsilon - x\|$, the associated analysis is termed a forward analysis. The analysis related to the residual error $\|y_\epsilon - y\|$ is the backward analysis, deferred to a later section.

We will now generalize the notion of x_ϵ to include a broader class of perturbations. For $x \in \mathbb{R}^n$ we write (2) as a series of n equations

$$x_i = f_i(y_\epsilon), \quad i = 1, \ldots, n \tag{11}$$

in which f_i denotes the i^{th} row of Φ. With the absolute error given by $\Delta x = \tilde{x} - x$, taken componentwise, the relative error at x_i, $\neq 0$, $i = 1, \ldots, n$ given by $r_{x_i} = \frac{\Delta x_i}{x_i}$ and assuming that f_i is regular (i.e., differentiable), each term in (10) can be given in further detail.

A. Arithmetic Stability and Conditioning

We consider now the general behavior of a perturbation Δx to the formulation Φ as it applies to the arithmetic stability. To first order, a differential sensitivity analysis give us

$$\begin{aligned} \Delta x_i &= \Phi(y_\epsilon) - \Phi(y) \\ &\approx \sum_{j=0}^{n} \frac{\partial f_i}{\partial y_j} \Delta y_j, \quad i = 1, \ldots, m \end{aligned} \tag{12}$$

or, in matrix notation

$$\Delta x = \begin{pmatrix} \Delta x_1 \\ \vdots \\ \Delta x_m \end{pmatrix} \approx \begin{pmatrix} \frac{\partial f_1}{\partial y_1} & \cdots & \frac{\partial f_1}{\partial y_n} \\ \vdots & & \vdots \\ \frac{\partial f_m}{\partial y_1} & \cdots & \frac{\partial f_m}{\partial y_n} \end{pmatrix} \begin{pmatrix} \Delta y_1 \\ \vdots \\ \Delta y_n \end{pmatrix} \tag{13}$$

$$= Df(y) \cdot \Delta y \tag{14}$$

with the operator D denoting the Jacobian matrix $Df(y)$. The quantities $\partial f / \partial y_j$ in (12) supply information regarding the amplification (or dampening) that can occur in the function

evaluation (mapping). The arithmetic stability involves the absolute error as given in equation (14). This is nothing more than the notion of stability given in Definition 5 when applied to the perturbations created from the computer representation of y.

To describe this in a relative sense we have, again to first order, that for $y_i \neq 0,\ i = 1, \ldots, n$

$$r_{x_i} \approx \sum_{j=0}^{n} \mathcal{K}_j(f_i) r_{y_j} \tag{15}$$

where the coefficients

$$\mathcal{K}_j(f_i) = \frac{y_j}{f_i(y)} \frac{\partial f_i(y)}{\partial y_j} \tag{16}$$

measure the magnification potential of the relative errors r_{y_j} and are commonly referred to as the condition numbers. When one or more of the factors are large, the problem (11) is said to be *ill-conditioned*. Otherwise, it is termed *well-conditioned*.

Definition 8: A problem (11) is *well-conditioned* if small perturbations to its input parameters do not cause drastic perturbations to its output parameter(s). Otherwise, it is an *ill-conditioned* problem.

For ill-conditioned problems a small perturbation in the input can cause large perturbations in the solution, independent of the arithmetic used in the computations (finite precision or exact).

The drawbacks from determining the condition number via (13) or (16) are mainly that

- the denominators $y_i,\ i = 1, \ldots, n$ must be nonzero, and
- $m \cdot n$ calculations are required.

When possible, one attempts to find a single (or at least fewer) number(s) to measure the conditioning of a problem. One such approach is to establish a relationship such as

$$\frac{\| f(\hat{y}) - f(y) \|}{\| f(y) \|} \leq \kappa \frac{\| \hat{y} - y \|}{\| y \|}. \tag{17}$$

As an example, we turn to numerical linear algebra. Wilkinson is attributed with establishing this connection using the concept of the condition number expressed in the form of equation (17): Let us assume that (1), the general problem of Section II, can be formulated as $Ax = b$, i.e., let $y = (a_1, \ldots, a_n, b)$. Here a_i are the columns to the nonsingular matrix A and $a_i, b \in \mathbb{R}^n$, for $1 \leq i \leq n$. Then $\Phi(y) = \Phi(A, b) = x$ if $\Phi(A, b) = A^{-1}b$. Due to restrictions, however, suppose we can only work with a nearby problem, $\Phi(\tilde{A}, \tilde{b}) = \tilde{x},\ \tilde{A} \cong A,\ \tilde{b} \cong b$.

In [4], one of his last publications, Wilkinson discussed this problem that was one of a number of problems hindering the research of algorithmic numerical stability in linear algebra. In the 1940's, a popular algorithm for solving this problem was Gaussian Elimination. At this time the elementary operations of this algorithm were orchestrated by individuals who relied on hand-cranked calculating machines for their evaluation. Gaussian Elimination was reported to have an inconsistent behavior, producing usable results for some problems and unusable results for others. As a consequence Gaussian Elimination was widely viewed as unstable (while, in light of the means used to compute its solution, Gaussian Elimination was very stable).

The paper of von Neumann and Goldstine [5] arrested most of the fears regarding the stability of Gaussian elimination, but to quote Wilkinson in [4], "... The reader is faced with page after page of detailed rounding error analysis and quite soon his senses are numbed. He may well emerge nursing the illusion that he has understood it in that he has followed the derivation of each line from the previous one, but there is more to 'understanding' than that". Wilkinson was not criticizing the important work of von Neumann and Goldstine, he was underscoring the importance of a systematic analysis in numerical linear algebra that included the effects of rounding errors as well as a problem's sensitivity to variations in the data. The difficulty with Gaussian Elimination was that occasionally an ill-conditioned problem would be solved. Wilkinson [6] was able to express this relationship in a simple manner:

Theorem 3: For A nonsingular, $\mathcal{K}(A) \equiv \| A \| \cdot \| A^{-1} \|$, and $\frac{\|A-\tilde{A}\|}{\|A\|} < \frac{1}{\mathcal{K}(A)}$, then $\tilde{A}$ is nonsingular and

$$\frac{\| x - \tilde{x} \|}{\| x \|} \leq \frac{\mathcal{K}(A)}{1 - \mathcal{K}(A)\frac{\|A-\tilde{A}\|}{\|A\|}} \left[\frac{\| A - \tilde{A} \|}{\| A \|} + \frac{\| b - \tilde{b} \|}{\| b \|} \right]. \tag{18}$$

Theorem 3 states that in general the relative error in the solution can be amplified by a large amount in comparison to the relative error in the problem. In this case, a large condition-number, $\mathcal{K}(A) \gg 1$, implies that the problem is ill-conditioned. Additionally, one might take

$$\kappa = \frac{\mathcal{K}(A)}{1 - \mathcal{K}(A)\frac{\|A-\tilde{A}\|}{\|A\|}}$$

to define the condition number but since it is often assumed for convenience that

$$\mathcal{K}(A)\frac{\| A - \tilde{A} \|}{\| A \|} \ll 1,$$

the condition number is defined as in the theorem.

B. Inherent Sensitivity

If ϵ quantifies machine precision, then one may describe δy through $\delta y = \epsilon \cdot \mid y \mid$. Applying equation (13), an assessment can then given to establish a threshold for the best that one may expect from the algorithm, i.e., after the input has been perturbed due to computer limitations but excluding the effects of roundoff resulting from computer arithmetic, a measurement is given to assess the sensitivity of the algorithm by which the computed solution can change. This can be seen as a separate but related measurement of the method's stability combined with the limitation to quantities representable by the computer. Stoer and Bulirsch [2, pg. 18] refer to this as the inherent error of the algorithm but we prefer to use the term inherent sensitivity.

Definition 9: Given an algorithm for solving the problem in (11), with $\mid \delta y \mid \leq \mid y \mid \epsilon$, and ϵ quantifying such factors as the machine roundoff, the *inherent sensitivity*, $\Delta_{\mathcal{I}} x$, of the computed solution, not taking into consideration the effects from computer arithmetic, is given by

$$\Delta_{\mathcal{I}} x := (\mid Df(y) \mid \cdot \mid y \mid + \mid x \mid) \epsilon. \tag{19}$$

Given that an algorithm is inherently-insensitive, one is now prepared to address the question of algorithmic reliability, i.e., to assess the effect from computer arithmetic. To accomplish this, the elementary operations (steps) of the algorithm must be taken into account as well as the arithmetic that is used for the computations. This will hence be dependent upon the ordering of the elementary operations as well as the computer in which the algorithm is housed.

C. Arithmetic Reliability

At each step of the method, each of which is not necessarily elementary, we define an operand set S containing all parameters, either the original input y, or the intermediate values resulting from previous steps. We may assume that whenever a parameter is no longer needed for future calculations it is deleted from S. The set S then describes how the vector of inputs y is transformed into the vector of outputs x. The size of S dynamically changes to reflect the inclusion of additional intermediate parameters and the deletion of input data or parameters that are no longer needed. Since there should be no confusion regarding the

intended meaning, the intermediate sets will, as in the case for algorithms, adopt a superscript enumeration scheme. At the i^{th}-step of the method we require n_i parameters to produce the n_{i+1} intermediate parameters needed for the next step. Hence, at step-i

$$f^i : S^i \longrightarrow \mathbb{R}^{n_{i+1}} \tag{20}$$

where

$$f^i(y^i) = y^{i+1}, \quad y^{i+1} = \begin{pmatrix} y_1^{i+1} \\ \vdots \\ y_{n_{i+1}}^{i+1} \end{pmatrix} \tag{21}$$

and $y^0 = y,\ y^r = x$. We note that by restricting each f^i to the representation of an elementary computer operation, we can in a natural manner transform the method into a complete characterization of the algorithm:

$$f(y) \quad = \quad f^r \circ f^{r-1} \circ \ldots \circ f^0(y^0) \tag{22}$$

in which

$$f^i : S^i \longrightarrow S^{i+1}, \quad i = 0, \ldots r-1, \quad S^i \subset \mathbb{R}^{n_i} \tag{23}$$

are elementary mappings and hence fully characterize the algorithm.

If F now denotes the implementation of method f (i.e., the algorithm), and ψ^i denotes the i^{th}-step plus all remaining steps in F, then referring to equation (22), each intermediate elementary operation can be analyzed to assess the roundoff error during that step of the calculation as well as its impact on the remaining calculations. This process, termed analyzing the *remainder map*, can be represented at this i^{th}-step by

$$\psi^i(y^i) = f^r \circ f^{r-1} \circ \ldots \circ f^i(y^i), \quad i = 0, 1, \ldots, r. \tag{24}$$

Here, y^i is defined as in (21). The interested reader is invited to [2, pp. 14-17] for an adequate number of examples.

If the roundoff errors from each step i of the algorithm are denoted as $\tau^i,\ 0 \leq i \leq r$, the effect of the input errors δy as well as the effects of the roundoff error on the computed solution $\tilde{x} = x + \delta x$ can be given, to first order, as

$$\delta x \quad = \quad \tilde{x} - x \tag{25}$$

$$\approx \underbrace{Df(y)\delta y}_{\mathcal{M}(\text{arith. stability})} + \underbrace{D\psi^1(y^1)\tau^1 + \ldots + D\psi^1(y^r)\tau^r + \tau^{r+1}}_{\mathcal{M}(\text{direct consistency})} \tag{26}$$

where $\tau^{r+1} \leq |\, x\,| \, \epsilon$ represents the roundoff effect after the last computation, i.e., the perturbation involved with the computer representation of the computed solution. To paraphrase, the total effect, δx, produced by the computer when solving a problem through a method implemented as an algorithm, can be partitioned to a part related to the arithmetic stability of the algorithm (independent of machine arithmetic or word-storage limitations) plus the direct consistency of the algorithm under the influence from machine arithmetic (dependent upon the sequencing of elementary operations, the machine arithmetic and the set of numbers representable by the computer).

To quantify what is regarded as acceptable with respect to the direct consistency term, we note that the inherent sensitivity expected from the unavoidable representation of data, specifically τ^{r+1}, gives us the lower bound, (direct consistency)$\geq \epsilon$.

Definition 10: An algorithm for solving (11) is *numerically reliable* if and only if *(direct consistency error)* $= O(\epsilon)$ at regular points.

Additionally, the inherent sensitivity of an algorithm can be used to establish a benchmark from which one can grade an algorithm: It would be unreasonable to expect the computed solution to have less uncertainty than that which can be expected from the conditioning of the problem, and in particular, from the inherent sensitivity due to the unavoidable variations in the input that one must expect from using a computer.

V. Backward Analysis

When attempting to estimate the error in the computed solution, the objective of a backward analysis is to transform the task concerned with computer effects to one that can make use of perturbation theory. This is accomplished by interpreting the computed solution $\tilde{x}$ as the exact solution to a nearby problem. This notion is in contrast to a forward analysis for a computed solution since for this the perturbations are not allowed to vary in order to fit $\tilde{x}$, instead they are given and specific to perturbations that are generated by the computer. This simple juxtapositioning of meaning has powerful consequences that can help tremendously with the analysis of an algorithm.

Given that there may be more than one nearby problem that $\tilde{x}$ solves exactly, we must specify which of these interests us. Additionally, we must specify what type of perturbations are allowed. This is important since the computed error estimate will only be accurate if the perturbations allowed for the problem are realistic with respect to the perturbations

expected through the implementation of the algorithm onto a computer. Let $\tilde{x} = \tilde{\Phi}(\hat{y})$ be the solution given by the formulation $x = \Phi(y)$ of $\phi(x) = y$ to solve the problem $\mathcal{P}$. If we allow perturbations that are in some class $\mathcal{E}$ and assume that if $\hat{y} = y + \Delta y,\ \Delta y \in \mathcal{E}$, then in exact arithmetic $\tilde{x} = \Phi(\hat{y})$. It may be the case that $\mathcal{E}$ is empty, but to proceed we will assume that the problem we are interested in has nontrivial elements $\Delta y \in \mathcal{E}$.

Definition 11: Let the problem $\mathcal{P}$ be given by $\phi(x) = y$ and formulated as $x = \Phi(y)$. For $\hat{y} = y + \Delta y$ and $\Delta y \in \mathcal{E}$, the *backward error* $B(\tilde{x})$ associated with the computed solution $\tilde{x}$ is given by

$$B(\tilde{x}) = \inf\{\|\Delta y\|; \Delta y \in \mathcal{E} \text{ and } \tilde{x} = \Phi(\hat{y})\}. \tag{27}$$

In this setting, the notion of arithmetic reliability, or more commonly backward stability, can now be formalized by relating the backward error to the machine precision, Ψ.

Definition 12: The cost function $J(\tilde{x})$ that measures the arithmetic reliability of an algorithm with respect to the computed solution $\tilde{x}$ is given by

$$J(\tilde{x}) = \frac{B(\tilde{x})}{\Psi}. \tag{28}$$

Definition 13: An algorithm is *backward stable*, or optimally reliable, when $J(\tilde{x}) = O(1)$.

Definitions 12 and 13 tell us that the most one can expect when running an algorithm on a computer is to get a computed solution that is in error of the order of the machine precision. If the data contains a significantly greater degree of uncertainty than Ψ, e.g., $y \leq \gamma\|y\|,\ \gamma >> \Psi$, then γ can replace Ψ in (28). To first order, we have forward error $\leq$ condition number $\times$ backward error. As discussed in Section IV-A, for the general condition number to be meaningfully defined we must restrict ourselves to regular problems, i.e., problems that are continuously differentiable.

The principles of forward and backward errors are complimentary: the Lax Principle quantifies the computability in finite precision and once demonstrated, the notion of backward stability is available to grade the quality of the algorithm with respect to either machine precision or the uncertainty in the data. Note that the condition number depends on the problem, i.e., $\phi(x) = y$ while the backward error is dependent upon the algorithm as well as the properties of the computer arithmetic. For linear systems we have Theorem 3 as an example.

VI. The Analysis of Multiple Algorithms

From the previous section it is possible for more than one implementation of a method to be reliable. When this is the case one then becomes interested in the performance of one algorithm relative to the other. It is clear that the term Df in (26) is independent of the implementation and hence the defining qualities of the algorithm. The remaining terms in (26), however, can be influenced by the organizational structure of the algorithm since the Jacobian matrices measuring the consistency depend on the remainder maps $D\psi^i$, i.e., on the expression

$$\mathcal{T}^1(\psi) = D\psi^1(y^1)\tau^1 + \ldots + D\psi^1(y^r)\tau^r. \tag{29}$$

Definition 14: Let $\mathcal{D}$ denote the domain of inputs for a given problem to be solved by some method. Furthermore, let algorithms φ and η be two different implementations of this method. For the evaluation of (29), algorithm φ is said to be *numerically more trustworthy* on $\mathcal{D}$ with respect to algorithm η if $|\mathcal{T}^1(\varphi)| < |\mathcal{T}^1(\eta)|$ for all $y \in \mathcal{D}$.

It should be noted, however, that in addition to the analysis of a complete algorithm, one can also meaningfully analyse a part of the algorithm that defines one or more elementary operations. This is useful when either a parameter of the algorithm has been identified as being particularly important to the computed solution, or when analysing all or part of a remainder map. One should recognize immediately from (26) that for such an analysis the inherent sensitivity of each elementary operation, considered separately, does not necessarily provide information regarding the inherent sensitivity of the entire algorithm. For example, an inherently insensitive algorithm may be defined by one or more elementary operations that, if analysed separately, are themselves inherently sensitive to machine arithmetic.

For the conditioning of matrices, a matrix A may be well conditioned but if one considers an algorithm for computing its LU decomposition and then separately analyses the steps that contribute to each factor, it is possible for each of L and U to be ill-conditioned while their product is well-conditioned.

A. The Direct vs. Indirect RLSL Algorithm

The implementation structure of the recursive least-squares lattice algorithm has been studied by Ling, Manolakis, and Proakis in [7] and [8]. Although no theoretical analysis could be found at the time to support their findings, it was nonetheless shown through heuristics

and various word length simulations that one implementation, termed the Direct version, was numerically more trustworthy than the other, termed the Indirect implementation. Since the greater majority of elementary operations were left unchanged, this work offers an example in which an analysis can be interested in only a portion of the algorithm(s). Here we discuss only the *a posteriori* version while in [3] both the *a priori* and the *a posteriori* versions were addressed, with similar modifications applying to each in order to transform the Indirect to the Direct algorithm.

Through algebraic manipulations the cross-correlation parameter $\Delta_{m+1}(n)$, one of the update parameters in the Indirect version, was eliminated from the algorithm. This was deemed significant since the manner in which $\Delta_{m+1}(n)$ was updated was demonstrated to be capable of manifesting large numerical error growth. Here, $m+1$ denotes the filter stage and n the time step. The update expression for $\Delta_{m+1}(n)$ is given by

$$\Delta_{m+1}(n) = \lambda\Delta_{m+1}(n-1) + \frac{b_m(n-1)f_m^*(n)}{\gamma_{m-1}(n-1)} \tag{30}$$

where $b_m(\cdot)$, $f_m^*(\cdot)$ and $\gamma_{m-1}(\cdot)$ denote the backward residual, the forward residual, and the likelihood parameter. Complex conjugation is denoted by $*$. The backward residual is a key parameter since it retains the information supplied in the input data and is used in place of it. The update expressions for $b_m(\cdot)$ (or $f_m(\cdot)$) require the use of their respective reflection coefficients, $\Gamma_{b,m+1}(n)$ (and $\Gamma_{f,m+1}(n)$, resp.):

$$\Gamma_{f,m+1}(n) = \frac{\Delta_{m+1}^*(n)}{\mathcal{F}_m(n)} \tag{31}$$

$$\Gamma_{b,m+1}(n) = \frac{\Delta_{m+1}(n)}{\mathcal{B}_{m-1}(n-1)} \tag{32}$$

where the forward and backward power is denoted by $\mathcal{F}_m(n)$ and $\mathcal{B}_{m-1}(n-1)$, respectively. Removing $\Delta_{m+1}(n)$ from the Indirect algorithm, but not the forward nor backward powers, introduces an interesting situation since each of these require a similar update structure and will be discussed further in the next subsection. The update expressions for the forward and backward reflection coefficients for the Direct algorithm become

$$\begin{aligned}\Gamma_{f,m+1}(n) &= \Gamma_{f,m+1}(n-1) - \frac{f_m(n)}{\mathcal{F}_m(n)\gamma_{m-1}(n-1)} \times \\ &\quad [b_m^*(n-1) + f_m^*(n)\Gamma_{f,m+1}(n-1)],\end{aligned} \tag{33}$$

and

$$\begin{aligned}\Gamma_{b,m+1}(n) &= \Gamma_{b,m+1}(n-1) - \frac{b_m(n-1)}{\mathcal{B}_m(n-1)\gamma_{m-1}(n-1)} \times \\ &\quad [f_m^*(n) + b_m^*(n-1)\Gamma_{b,m+1}(n-1)],\end{aligned} \tag{34}$$

respectively. Additionally, there is a similar exchange for the update expression for the filter-, or ladder-, gain coefficient that we will not give explicitly here. We can compare these related, but different algorithms through the use of the preceding sections.

From (29), if the equation exchanges occur at the i^{th} and $(i+1)^{st}$ elementary steps, the remainder map at step i for the Direct and Indirect can be compared. Analyzing this and all subsequent remainder maps allows us to focus on the arithmetic effects due to the different manner that each algorithmic variant updates its reflection coefficients.

A similar approach is to focus on the update equations for the forward and backward residuals that are common to both algorithms:

$$f_m(n) = f_{m-1}(n) + \Gamma^*_{f,m}(n) b_{m-1}(n-1) \tag{35}$$

$$b_m(n) = b_{m-1}(n-1) + \Gamma^*_{b,m}(n) f_{m-1}(n). \tag{36}$$

Since the differences in the algorithms ultimately lie in the manner in which the reflection coefficients are updated, one can apply (29) to these equations and see that although their inherent sensitivity will be identical, the perturbations of $\Gamma^*_{f,m}(n)$ and $\Gamma^*_{b,m}(n)$ will depend on the ordering of elementary operations, and hence will be different. One can conclude, as was concluded in [9], that although the Direct variant may be numerically more trustworthy with respect to the Indirect variant, both algorithms can produce unusable or poorly computed results due to their shared inherent sensitivity to perturbations (that is common to all implementations of a lattice-based formulation).

B. The Conditioning of the Lattice-Based Method

If we consider (35) and (36) as two elementary operations of the lattice method, we can interpret it as the following composition of functions

$$\phi(y) = \phi^1 \circ \phi^0(y^0) \tag{37}$$

$$= \phi^1(y^1) \tag{38}$$

$$= \begin{pmatrix} f_m(n) \\ b_m(n) \end{pmatrix} \tag{39}$$

in which ϕ^0 is defined through (35) and ϕ^1 through (36). The initial and intermediate input y^0 and y^1, respectively, is given by

$$y^0 = \begin{pmatrix} f_{m-1}(n) \\ b_{m-1}(n-1) \\ \Gamma^*_{f,m}(n) \\ \Gamma^*_{b,m}(n) \end{pmatrix} \tag{40}$$

and

$$y^1 = \begin{pmatrix} f_m(n) \\ f_{m-1}(n) \\ b_{m-1}(n-1) \\ \Gamma^*_{f,m}(n) \\ \Gamma^*_{b,m}(n) \end{pmatrix}. \tag{41}$$

For simplicity, if we assume that the residuals are real-valued scalars, and in light of (20) through (23), we have that $S^0 \longrightarrow \mathbb{R}^4, S^1 \longrightarrow \mathbb{R}^5$, and $\phi^1(y^1) \subset S^3 \longrightarrow \mathbb{R}^2 \supset \phi(y)$. The relative errors $r_\phi = (r_{f,m}(n), r_{b,m}(n))^t$ for the forward and backward residuals, given perturbations δy in the input as described by (40), is given by

$$r_\phi = \begin{pmatrix} {}^r f_m(n) \\ {}^r b_m(n) \end{pmatrix} \tag{42}$$

$$= \left(\begin{array}{c|c|c|c} \mathcal{K}_1 & \mathcal{K}_3 & \mathcal{K}_3 & 0 \\ \hline \mathcal{K}_4 & \mathcal{K}_2 & 0 & \mathcal{K}_4 \end{array}\right) \begin{pmatrix} {}^r f_{m-1}(n) \\ {}^r b_{m-1}(n-1) \\ {}^r \Gamma_{f,m}(n) \\ {}^r \Gamma_{b,m}(n) \end{pmatrix}, \tag{43}$$

where, $\mathcal{K}_1 = \frac{f_{m-1}(n)}{f_m(n)}$, $\mathcal{K}_2 = \frac{b_{m-1}(n-1)}{b_m(n)}$, $\mathcal{K}_3 = \frac{b_{m-1}(n-1)\ \Gamma_{f,m-1}(n)}{f_m(n)}$ and $\mathcal{K}_4 = \frac{f_{m-1}(n)\ \Gamma_{b,m}(n)}{b_m(n)}$. The condition numbers in this case are given explicitly for each parameter. In Table I we show how the condition numbers will effect each of the relative errors that are present in the parameters used to update the forward and backward residuals. A $\surd$ in row i and column j signifies that the relative error denoted in the first column of row i will be multiplied by the condition number given at the top of column j. This is in contrast to the manner in which the conditioning of equations (35) and (36) was formulated in [9] through a single representative condition number of a 2×2 matrix.

TABLE I

CONDITION NUMBERS $\mathcal{K}_i,\ i = 1, \ldots, 4$ AMPLIFYING/DAMPENING RELATIVE ERRORS IN THE UPDATE PARAMETERS SHOW IN PARTICULAR HOW RELATIVE ERRORS IN THE REFLECTION COEFFICIENTS PROPAGATE INTO THE FORWARD AND BACKWARD RESIDUALS.

	Condition Numbers			
	$\mathcal{K}_1$	$\mathcal{K}_2$	$\mathcal{K}_3$	$\mathcal{K}_4$
Relative Errors				
${}^{r}f_{m-1}(n)$	$\surd$			$\surd$
${}^{r}b_{m-1}(n-1)$		$\surd$	$\surd$	
${}^{r}\Gamma_{f,m}(n)$			$\surd$	
${}^{r}\Gamma_{b,m}(n)$				$\surd$

$\mathcal{K}_1 = \dfrac{f_{m-1}(n)}{f_m(n)}$, $\mathcal{K}_2 = \dfrac{b_{m-1}(n-1)}{b_m(n)}$, $\mathcal{K}_3 = \dfrac{b_{m-1}(n-1)\ \Gamma_{f,m}(n)}{f_m(n)}$,

$\mathcal{K}_4 = \dfrac{f_{m-1}(n)\ \Gamma_{b,m}(n)}{b_m(n)}$

Of interest as well is the possible inference that the better performance of the Direct version to the Indirect is the result of the deletion of the cross-correlation parameter given in (30). At first glance it might be thought that any errors in $\Delta_{m-1}(n-1)$ will be damped out exponentially by the factor λ. In fact, this is the basis for the results in the analysis that will be reviewed in the following section. Here, however, one can find that the errors may grow in time, as shown in [10] for the forward and backward power update recursions (that share the same update structure as (30)). Since the forward and backward power recursions are present in the Direct version of the RLSL algorithm, it must be that error dampening occurs when parameters require either the forward or backward power. Error dampening is the terminology applied to the situation in which the computed result from an elementary operation carries a small relative error after one or more of the involved parameters was associated with a large relative error.

VII. The Literature Revisited

To discriminate between a well-conditioned problem, a stable method and a stable algorithm, there must exist a clear ideological separation between the three that begins with the terminology. We have emphasized this point in the previous sections and now we will illustrate it through a previous analysis that is taken from the literature. Specifically, we revisit the important work of S. and L. Ljung [3]. In this work the authors analyzed conventional least squares algorithms, the fast least squares algorithm, and the fast lattice algorithms. It was proved that the conventional least squares algorithms and the fast lattice algorithms are exponentially stable in the sense of Lyapunov, i.e., the effect of an error in the input state decays exponentially. Conversely, the fast least squares algorithm was shown to be unstable with respect to a perturbed input state.

From this statement, however, it is left unclear whether the stable algorithms can be safely implemented and run continuously on a computer because their analysis used notions and terminology for conditioning, inherent sensitivity, and stability interchangeably. As will be seen, the actual analysis performed in [3] pertains to the arithmetic stability which can now be distinguished from the notion of arithmetic computability and arithmetic reliability.

First, by introducing their error notation for the input in the context of machine effects, i.e., $\tilde{\xi}(t) = \xi(t) + \delta\xi$ and $\tilde{x}(t) = x(t) + \delta x$ where $\delta\xi$ and δx pertain to arithmetic effects such as representation errors and machine arithmetic, Ljung and Ljung implied an analysis that applied to either the arithmetic computability of the solution as described in equation (9), or an inherent sensitivity analysis as described in equation (19). However, later their analysis grouped together algorithms that were mathematically equivalent for the entire domain of persistently excited inputs for the problem. By doing this their analysis could not assess a measurement of arithmetic computability or reliability, but it could only pertain to the arithmetic stability of the algorithm. Their use of the term stability can now be made unequivocal and not confused with considerations dependent upon the implementation that this term might otherwise have implied when perturbations are defined with respect to machine characteristics. Although their results classified the stability (exponential in the sense of Lyapunov), we will state it in a more general context to establish its relation with respect to the other possibilities given in the previous sections.

Theorem 4: The conventional LS algorithm, including Bierman's UD-factorization algo-

rithm and the fast lattice algorithm as given in [3] are based upon *arithmetically stable* algorithms.

For the Fast Least Squares algorithm, an analysis involving (14) was performed for a special choice of the input vector. For this the vector y involved the four parameters of the state of the filter as well as the input data at times t and $t-1$. The function $f(y)$ was differentiated with respect to the state y and to eliminate the parameter of time from the matrix elements, a product was taken of the Jacobian at two successive time measurements. Since at least one of the eigenvalues corresponding to this matrix product was shown to be larger than unity in modulus, the algorithm, through this example, was shown to be arithmetically unstable. Any perturbation in the direction of the eigenvector associated with the largest eigenvalue will be amplified, and eventually, due to the iterative nature of the algorithm, dominate the filter solution.

Theorem 5: The fast least squares, or fast Kalman, algorithm is *arithmetically unstable.*

We have illustrated how an existing framework (e.g., [2], [1]) can be applied to analyses in signal processing to facilitate the unambiguous and correct interpretation of results. Stability, if allowed to remain a general term, covers a spectrum of meanings whose breadth invites confusion.

VIII. Conclusion

The techniques and results of many fields that have been utilized when analyzing the numerical properties of signal processing algorithms often involve exact as well as finite precision arithmetic when defining stability. We have found that this offers a singular source for confusion when one is pondering the ultimate question regarding the usefulness of a solution produced by an algorithm whose calculations are performed in finite precision arithmetic. This understanding has motivated definitions that make a clear distinction between the sources of error that, with respect to exact arithmetic, are inherent to the problem (conditioning), and the stability and consistency of the formulation of the problem (the method). With respect to inexact arithmetic, definitions are in place to classify the computability of an algorithm through the notions of arithmetic stability and reliability.

A user of an algorithm desires a guarantee that small perturbations will remain small. By considering the behavior of a perturbation as it tends to zero, the notion of convergence has been presented. Convergence in this sense means that the perturbed value becomes exact

as the perturbation tends to zero. The perturbations can occur in the function f or in its parameter y. Each have been addressed separately and given its own terminology: The former involves stability while the later involves consistency. Additionally, the type of arithmetic, exact or finite, further delineates the perturbation; the terms given above apply to exact arithmetic while for finite precision arithmetic we have arithmetic stability replacing stability and arithmetic reliability replacing the term consistency. For a solution determined from an algorithm that computed its results in finite precision (computer) arithmetic, small perturbations will remain small if the algorithm is arithmetically stable and arithmetically reliable. To guarantee that an algorithm will produce good results when its solution is computed in exact arithmetic, one must show that the algorithm is stable and consistent. The techniques vary widely, however, the objective remains unchanged. To assist, the notion of conditioning exists as well as the backwards analysis technique.

To make this issue clear we have reviewed the same terms as applied to numerical linear algebra as well as the current progress for the analysis of signal processing algorithms. Applying these terms to the work of Ljung and Ljung in [3] and Bunch et al. in [9], we offered examples for the unambiguous use of these terms.

References

[1] F. Chaitin-Chatelin and V. Frayssé, *Lectures on Finite Precision Computations*, SIAM, Philadelphia, 1996.

[2] J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, Springer-Verlag, New York, Heidelberg, Berlin, 1983.

[3] S. Ljung and L. Ljung, "Error propagation properties of recursive least-squares adaptation algortithms," *Automatica*, vol. 21, no. 2, pp. 157–167, 1985.

[4] J.H. Wilkinson, "Error analysis revisited," *IMA Bulletin*, vol. 22, pp. 192–200, 1986.

[5] J. von Neumann and H. Goldstine, "Numerical inverting of matrices of high order," *Bull. Amer. Math. Soc.*, vol. 53, pp. 1021–99, 1947.

[6] J.H. Wilkinson, "Error analysis of direct methods of matrix inversion," *J. Assoc. Comput. Mach.*, vol. 8, pp. 281–330, 1961.

[7] F. Ling, D. Manolakis, and J.G. Proakis, "New forms of LS lattice algorithms and an analysis of their round-off error characteristics," in *Proc. ICASSP*, Tampa, FL, 1985, pp. 1739–1742.

[8] F. Ling, D. Manolakis, and J.G. Proakis, "Numerically robust least-squares lattice-ladder algorithms with direct updating of the reflection coefficients," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. ASSP-34, no. 4, pp. 837–845, Aug. 1986.

[9] J. R. Bunch, R. C. Le Borne, and I. K. Proudler, "Tracking ill-conditioning for the RLS-lattice algorithms," *IEE Proceedings Vision, Image and Signal Processing*, vol. 145, no. 1, pp. 1–6, Feb. 1998.

[10] J. R. Bunch and R. C. Le Borne, "Error accumulation effects for the a posteriori RLSL prediction filter," *IEEE Transactions on Signal Processing*, vol. 43, 1995.