Reliable Computing 2 (3) (1996), pp. 219–228

Fast error estimates for indirect measurements: applications to pavement engineering

Carlos Ferregut, Soheil Nazarian, Krishnamohan Vennalaganti,
Ching-Chuan Chang, and Vladik Kreinovich

In many real-life situations, we have the following problem: we want to know the value of some characteristic y that is difficult to measure directly (e.g., lifetime of a pavement, efficiency of an engine, etc.). To estimate y, we must know the relationship between y and some directly measurable physical quantities $x_1, \ldots, x_n$. From this relationship, we extract an algorithm f that allows us, given x_i, to compute y: $y = f(x_1, \ldots, x_n)$. So, we measure x_i, apply an algorithm f, and get the desired estimate.

Existing algorithms for error estimate (interval mathematics, Monte-Carlo methods, numerical differentiation, etc.) require computation time that is several times larger than the time necessary to compute $y = f(x_1, \ldots, x_n)$. So, if an algorithm f is already time-consuming, error estimates will take too long.

In many cases, this algorithm f consists of two parts: first, we use x_i to determine the parameters z_k of a model that describes the measured object, and second, we use these parameters to estimate y. The most time-consuming part is finding z_k; this is done by solving a system of non-linear equations; usually least squares method is used.

We show that for such f, one can estimate errors repeating this time-consuming part of f only once. So, we can compute both y and an error estimate for y with practically no increase in total computation time. As an example of this methodology, we give pavement lifetime estimates.

Быстрое оценивание погрешностей при непрямых измерениях: приложения к проектированию мостовых

К. Феррегут, С. Назарян, К. Венналаганти, Ч.-Ч. Чанг, В. Крейнович

На практике часто встречается следующая задача: требуется определить значение некоей характеристической переменной y, которую трудно измерить напрямую (например, срок жизни мостовой, эффективность двигателя и т.п.). Чтобы оценить величину y, мы должны знать соотношение между y и некоторыми физическими величинами $x_1, \ldots, x_n$, которые подлаются прямому измерению. На основании этого соотношения строится алгоритм f, который позволяет исходя из заданных x_i вычислить y: $y = f(x_1, \ldots, x_n)$. После этого останется лишь измерить x_i, применить алгоритм f и получить требуемую оценку.

Существующие алгоритмы оценивания погрешностей (интервальная математика, методы Монте-Карло, численное дифференцирование и т. п.) требуют затрат времени, в несколько раз

This work was supported by NSF grants No. CDA–9015006 and IRI–92–11–662, NASA Research Grant No. 9–482, and a grant from the Materials Research Institute. We want to thank the United States Army Corps of Engineers, Waterways Experiment Station (USAE, WES), Vicksburg, MS, for supplying us with the Layered Elastic Programs, WELSEA and WESDEF. We are also greatly thankful to all the participants of the International Conference on Numerical Analysis with Automatic Result Verification, Lafayette, LA, February–March 1993, especially to Drs. S. Markov and W. Walster, for valuable discussions.

превышающих те, которые нужны для вычисления $y = f(x_1, \ldots, x_n)$. Поэтому, если алгоритм f и без того работает медленно, оценивание погрешностей может занять неприемлемо большое время.

Во многих случаях алгоритм f состоит из двух частей: сначала на основе x_i определяются параметры z_k модели, описывающей измеряемый объект, а затем эти параметры используются для оценки y. Наибольших временных затрат требует получение значений z_k, для чего необходимо решить систему нелинейных уравнений (как правило, применяется метод наименьших квадратов).

Показано, что для подобных алгоритмов f можно получить оценки погрешностей, произведя эту трудоемкую часть вычислений только один раз. Таким образом, мы можем вычислить как y, так и оценку погрешности для y практически без увеличения общего времени вычислений. В качестве примера использования этой методики приводятся оценки срока жизни мостовой.

1. Formulation of a general problem: error estimation for indirect measurements

Indirect measurements: a real-life situation. In this paper, we will consider the following engineering problem:

We want: to know the value of some physical quantity y.

Problem: It is difficult (or even impossible) to measure y directly (examples of such quantities are lifetime of a pavement, efficiency of an engine, etc.).

Solution: To estimate y, we must know the relationship between y and some directly measurable physical quantities $x_1, \ldots, x_n$. From this relationship, we extract an algorithm f that allows us, given x_i, to compute y: $y = f(x_1, \ldots, x_n)$. So, we measure x_i, apply an algorithm f to the measurement results $\tilde{x}_i$, and get the desired estimate $\tilde{y} = f(\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n)$. This procedure is called an *indirect measurement* of y.

Warning: This f is rarely an explicit analytical expression; usually, it is a complicated algorithm.

Remaining problem: What is the precision of the resulting value $\tilde{y}$?

What do we know about the errors of measuring x_i? The error of an indirect measurement is caused by the errors with which we know x_i. So, to estimate the precision of y, let us find out what we know about these precisions.

For each measuring device, its supplier must provide some information about its precision (else, if no precision is guaranteed, this measuring device is of no use). There are two main ways to describe such a precision (see, e.g., [4]):

1) In many cases, we know only a characteristic of the *total* error $\Delta x_i = \tilde{x}_i - x_i$. Namely, the supplier provides a number Δ_i such that the error never exceeds Δ_i ($|\Delta x_i| \leq \Delta_i$); in other words, the actual value x_i belongs to an interval $[\tilde{x}_i - \varepsilon, \tilde{x}_i + \varepsilon]$.

This number Δ_i can depend on $\tilde{x}_i$. For example, a supplier can provide us with a *relative* error δ_i, meaning that $|\Delta x_i| \leq \delta_i \tilde{x}_i$ (so, in this case, $\Delta_i = \delta_i \tilde{x}_i$).

2) Measurement error Δx is random. Therefore, we can represent it as a sum of two components: the mathematical expectation $E(\Delta x)$ that is called a *systematic error* (denoted by Δx_i^s), and the difference $\Delta x - E(\Delta x)$ that is called a *random error* (and denoted by Δx_i^r). For some measuring devices, we know the characteristics of the both components. Namely, the supplier provides a value Δ_i^s such that $|\Delta x_i^s| \leq \Delta_i^s$, and a value σ_i such that the standard deviation $\sigma(\Delta x_i)$ does not exceed σ_i.

These two characteristics can also depend on $\tilde{x}_i$. For example, they can be given as *relative* errors δ_i^s and s_i. In this case, $\Delta_i^s = \tilde{x}_i\delta_i^s$ and $\sigma_i = \tilde{x}_i s_i$.

Systematic errors are usually negligible. To improve the quality of a measuring device, it makes sense to *calibrate* it first. By this we mean that before we release this device, we compare its results with the results of a more precise device, find an average error, and compensate for it. For calibrated devices, the systematic error is negligible, so we can assume that the total error coincides with its random component. After the calibration, we also have enough data to estimate the standard deviation $\sigma(x_i)$. So, we can assume that $\sigma(x_i)$ is known.

In view of this, in this paper, we will consider only two cases:

1) we known an interval for an error;

2) the error has 0 average, we know its standard deviation $\sigma(x_i)$, and the errors of different measurements are independent random variables.

2. How are errors estimated now?

The main methodologies for estimating the error of y are: numerical differentiation, Monte-Carlo method (for detailed description of these methods, see, e.g., [1, 2, 5, 13–15]), and interval mathematics (see, e.g., [11]). Let us briefly describe these methods.

Numerical differentiation. The error $\Delta y = \tilde{y} - y$ can be expressed as the difference

$$f(\tilde{x}_1, \ldots, \tilde{x}_n) - f(x_1, \ldots, x_n) = f(\tilde{x}_1, \ldots, \tilde{x}_n) - f(\tilde{x}_1 - \Delta x_1, \ldots, \tilde{x}_n - \Delta x_n). \tag{1}$$

Usually, the errors Δx_i are relatively small, so we can neglect the terms that are quadratic in errors. If we neglect quadratic (and higher order terms) in the Taylor expansion of the r.h.s. of (1), we can conclude that

$$\Delta y = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}\Delta x_i. \tag{2}$$

If we know that $|\Delta x_i| \le \Delta_i$, then we can conclude that $|\Delta y| \le \Delta$, where

$$\Delta = \sum_{i=1}^{n} \left|\frac{\partial f}{\partial x_i}\right|\Delta_i. \tag{3}$$

If we know $\sigma(x_i)$, then we can compute $\sigma(y)$ as

$$\sigma(y) = \sqrt{\sum_{i=1}^{n} \left(\frac{\partial f}{\partial x_i}\right)^2 \sigma^2(x_i)}. \tag{4}$$

Since f is usually given as an algorithm, and not as an analytical expression, we must somehow compute the partial derivatives. We can do that by applying formula (2) to the case when $\Delta x_i = h$ for some i and 0 for all other i (here, h is some small number). As a result, we get the following estimate:

$$\frac{\partial f}{\partial x_i} = \frac{f(\tilde{x}_1, \ldots, \tilde{x}_{i-1}, \tilde{x}_i, \tilde{x}_{i+1}, \ldots, \tilde{x}_n) - f(\tilde{x}_1, \ldots, \tilde{x}_{i-1}, \tilde{x}_i - h, \tilde{x}_{i+1}, \ldots, \tilde{x}_n)}{h}. \tag{5}$$

This is a standard formula for numerical differentiation.

So, in order to get an estimate for Δy, we must compute the partial derivatives using (5), and then estimate Δ or $\sigma(y)$ using (3) or (4).

Monte-Carlo methods. A standard Monte-Carlo method (see, e.g., [13]) is used to estimate $\sigma(y)$. This method is based on the following idea: First, we choose the number of iterations N. Then, for $k = 1, \ldots, N$, we substitute into f the values $\tilde{x}_i + \beta_i^k$, where β_i^k are computer-simulated normally distributed random numbers with 0 average and standard deviation $\sigma(x_i)$. As a result, we get N values $y^1, \ldots, y^N$. Because of (2), the distribution of $y^k - y$ is Gaussian, with 0 average and standard deviation $\sigma(y)$. So, we can estimate $\sigma(y)$ as $\sqrt{\sum(y^k - y)^2/N}$.

This method gives an approximate value of $\sigma(y)$. In real life, it is sufficient to know $\sigma(y)$ with $\approx 20\%$ precision (practically no one distinguishes between the cases when the precision is, say, 0.01, or 0.012, which is 20% more). To get such precision, we need $N \approx 25$ iterations.

A Monte-Carlo style method to compute Δ was proposed in [7] (see also [6]). This method uses $N \approx 50$ iterations to guarantee the desired 20% precision.

Interval techniques. Standard techniques of interval mathematics require that we decompose the algorithm f into elementary operations, and then apply interval operations instead of usual ones.

3. What is wrong with the existing methods

To use a numerical differentiation method, we must apply the algorithm f $n + 1$ times: once to get $\tilde{y}$, and then n times to estimate partial derivatives.

For example, if we have 10 variables, we need 11 times more time to estimate the error than to compute y itself. If f is fast, there is no problem. But **if f is already a time-consuming algorithm, this is no good.**

For Monte Carlo methods, we must apply f 25 or 50 times; so, the computation time for this method is 25–50 times longer than the time that is necessary to compute y.

For interval methods: an operation with intervals consist of 2 or 4 operations with numbers. Therefore, by applying interval mathematics, we increase the computation time at least 2–4 times. In addition, the error estimates that we obtain this way are often "overshoots" (i.e., much bigger than the biggest possible errors).

How to diminish this computation time? If we have several processors that can work in parallel, then we can use parallel algorithms for a speed-up (see, e.g., [6]). But what if we have only one processor?

In this paper, we will describe a frequent real-life situation, when we can drastically decrease this computation time.

4. The new method of estimating errors

To develop a new method, we will use the fact that in many real-life situations, the algorithm f is of a very specific type, and for such f, we can estimate the error in y really fast.

To elaborate on that, we need to recall where f usually comes from.

Where does f come from? We have mentioned that an algorithm f comes from the known relationship between x_i and y. Usually, such a relationship exists in a form of a *model* of the physical phenomenon. By a model, we mean that we have (relatively simple) formulas

or algorithms that, given some parameters, allow us to compute all physical characteristics, including x_i and y.

Some of these parameters can be known from the previous experiments (we will denote them by $t_1, \ldots, t_q$). Some of these parameters are specific to this particular situation (we will denote them by $z_1, \ldots, z_p$). These paremeters must be determined from the known measurement results $\tilde{x}_1, \ldots, \tilde{x}_n$.

So, an algorithm that estimates $\tilde{y}$ from $\tilde{x}_i$, consists of *two stages*:

- *first*, we use the measurement results $\tilde{x}_i$ to get estimates $\tilde{z}_k$ for the parameters of the model, and
- *second*, we use the estimates $\tilde{z}_k$ and $\tilde{t}_l$ to estimate y.

Let us denote by F_i and F functions that describe the model, i.e., that express x_i and y in terms of z_k and t_l: $x_i = F_i(z_1, \ldots, z_p, t_1, \ldots, t_q)$ and $y = F(z_1, \ldots, z_p, t_1, \ldots, t_q)$. Then, to estimate z_k, we must solve the system of equations $x_i = F_i(z_1, \ldots, z_p, t_1, \ldots, t_q), 1 \le i \le n$. Since we know only approximate values of x_i, we will get approximate values of z_k as well. To improve the precision, we can perform additional measurements. In this case, we will get an *over-determined system* (with more equations than unknowns). Usually, such systems are solved by the least-squares method, i.e., from the condition that

$$\sum_{i=1}^{n} \left(F_i(z_1, \ldots, z_p, t_1, \ldots, t_q) - x_i\right)^2 \to \min_{z_1, \ldots, z_p}.$$

Example. For example, in celestial mechanics, there are rather simple formulas that describe a trajectory of any celestial body in a Solar system (e.g., a comet). The parameters of this trajectory include the position t_i of the Sun (that is well known from other measurements), and parameters z_k of the orbit relative to the Sun (that have to be determined from the observations). So, if we want to predict a future position y from several observations $\tilde{x}_i$, we must find the parameters z_k of a trajectory from $\tilde{x}_i$, and then predict y.

What is the most time-consuming part of this algorithm? If F_i are linear, we can easily solve a system of linear equations and find z_k. The problem becomes computationally complicated when the functions F_i are highly non-linear. In this case, no matter what method we use to compute z_k (Newton's method, other optimization techniques, etc.), all these methods consist of several iterations: we find a first guess $\vec{z}$, substitute these values into F_i, then adjust the values of z_k, etc. In other words, the first "back-calculation" stage (estimating z_k from $\tilde{x}_i$) includes several "forward" steps (computing x_i from z_k). Therefore, the computation time for the first stage is much longer than the time for the second stage.

So, *the most time-consuming part of an algorithm f is estimaing z_k from $\tilde{x}_i$.*

Our idea. The reason why existing methods take so long is that they involve several applications of f and, therefore, several repetitions of a time-consuming "back-calculation" step: for the initial values $\vec{\tilde{x}} = (\tilde{x}_1, \ldots, \tilde{x}_n)$, to get the estimates $\tilde{z}_k$, and then, for several other input values $\vec{x}$ that are close to $\tilde{x}_i$.

Of course, we need to apply this back-calculation once, to get the values $\tilde{z}_k$. But next time, when we want to compute z_k for some close values x_k, we do not have to repeat this complicated procedure of solving a system of non-linear equations: indeed, if x_i are close to $\tilde{x}_i$, then the corresponding z_k will be close to $\tilde{z}_k$. So, it is sufficient to find the small deviations $\Delta z_k = z_k - \tilde{z}_k$. We consider the case when the errors are relatively small, so we can neglect

the terms that are quadratic in Δz_k, and therefore, instead of the original non-linear system of equations $F_i(\vec{z}, \vec{t}) = \tilde{x}_i$, solve its linearized version.

How to use this idea in case of interval estimates. Since we assume that our model is accurate, the true values z_k and t_l of the parameters must satisfy the equations $F_i(z_1, \ldots, z_p, t_1, \ldots, t_q) = x_i$.

If we substitute $z_k = \tilde{z}_k - \Delta z_k$, $t_l = \tilde{t}_l - \Delta t_l$, and $x_i = \tilde{x}_i - \Delta x_i$ into these equations, expand F_i into Taylor series, and retain only terms that are linear in Δz_k and Δt_l, we will conclude that

$$\sum_k a_{ik} \Delta z_k = \Delta x_i - \delta_i - \sum_l b_{il} \Delta t_l \tag{6}$$

where $\delta_i = \tilde{x}_i - F_i(\tilde{z}_k, \tilde{t}_l)$,

$$a_{ik} = \frac{\partial F_i}{\partial z_k}, \quad b_{il} = \frac{\partial F_i}{\partial t_l}. \tag{7}$$

From $y = F(z_1, \ldots, z_p, t_1, \ldots, t_q)$, we can likewise conclude that

$$\Delta y = \sum_k c_k \Delta z_k + \sum_l d_l \Delta t_l \tag{8}$$

where

$$c_k = \frac{\partial F}{\partial z_k}, \quad d_l = \frac{\partial F}{\partial t_l}. \tag{9}$$

The only thing we know about the values Δx_i, Δz_k and Δt_l is that they satisfy (6) and that the values of Δx_i and Δt_l are limited by the corresponsing measurement errors. So, to find possible the interval of possible values of Δy, we must find the biggest and the smallest value of Δy under these conditions. Since these condiditons are linear in the unknowns, this optimization problem becomes a well-known linear programming problem, for whicn fast algorithms are well-known.

So, we are ready to formulate an algorithm.

Algorithm that estimates errors for the case when we know intervals.

Given:

1) An algorithm F_i that estimates x_i from z_k and t_l.
2) An algorithm F that estimates y from z_k and t_l.
3) An algorithm f that gives an estimate $\tilde{y}$ for y from $\tilde{x}_i$ by first estimating z_k and then estimating y.
4) The measured values $\tilde{x}_i, 1 \leq i \leq n$ and their precisions Δ_i (i.e., such numbers that $|\Delta x_i| = |\tilde{x}_i - x_i| \leq \Delta_i$).
5) The measured values $t_1, \ldots, t_q$, and their precisions Δ_l^t.

Objective: To find an estimate Δ for $\Delta y = y - \tilde{y}$ (i.e., a value Δ such that $|\Delta y| \leq \Delta$).

Algorithm:

1) Apply f and get estimates $\tilde{z}_k$ and $\tilde{y}$.

2) Use numerical differentiation to estimate the derivatives (7) and (9).

3) Estimate $\delta_i = \Delta x_i - F_i(\tilde{z}_1, \ldots, \tilde{z}_p, \tilde{t}_1, \ldots, \tilde{t}_q)$.

4) Find Δ^+ as a solutions of the following linear programming problem with unknowns Δx_i, Δz_k, and Δt_l: $\sum c_k \Delta z_k + d_l \Delta t_l \to \max$ under the conditions (6), $-\Delta_i \le \Delta x_i \le \Delta_i$ $(1 \le i \le n)$, and $-\Delta_l^t \le \Delta t \le \Delta_l^t (1 \le l \le q)$.

5) Find Δ^- as a solution to a similar problem, but with min instead of max.

6) Compute $\Delta = \max(\Delta^+, |\Delta^-|)$.

Comment. In this algorithm, a time-consuming "back-calculation" part is repeated only once. Therefore, for time-consuming f, this algorithm enables us to estimate Δy with practiaclly no increase in total computation time.

The case of random errors. In this case, for x_i close to $\tilde{x}_i$, we can also linearize the expression for F_i. Threfore, instead of a system of non-linear equations, we get a linearized system (6). After this system has been solved, we can use (8) to estimate Δy.

We must estimate the standard deviation of y. Because of (8),

$$\sigma(y)^2 = E(\Delta y)^2 = \sum c_k c_{k'} E(\Delta y_k \Delta y_{k'}) + \sum c_k d_l E(\Delta y_k \Delta t_l) + \sum d_l d_{l'} E(\Delta t_l \Delta t_{l'}). \quad (10)$$

So, to estimate $\sigma(y)$, we must estimate these mathematical expetations $E(\Delta y_k \Delta y_{k'})$, etc. If we denote $X_k = \Delta z_k$ and $X_{p+l} = \Delta t_l$, then can say that we must know $E(X_k X_l)$ for all k and l.

To get these estimates, we can apply the standard methodology of (linear) least squares method (see, e.g., [8, 10]). We are interested in $p+q$ variables $\Delta z_k, 1 \le k \le p$ and $\Delta t_l, 1 \le l \le q$. For these unknowns, we have the following system of equations: $\sum a_{ik} \Delta z_k + \sum b_{il} \Delta t_l \approx 0$ with standard deviation $\sigma(x_i)$, and $\Delta t_l \approx 0$ with standard deviation $\sigma(t_l)$. In terms of X_k, we have $n + q$ equations: $\sum A_{ik} X_k \approx 0$ with standard deviation σ_i, where for $i \le n$: $A_{ik} = a_{ik}$ for $k \le p$, and $A_{i,p+l} = b_{pl}$; for $i > n$, $A_{n+l,l} = 1$ and $A_{n+l,k} = 0$ for $k \ne n + l$; $\sigma_i = \sigma(x_i)$ for $i \le n$, and $\sigma_{n+l} = \sigma(t_l)$.

According to the least squares theory, the matrix $E(X_k X_l)$ is an inverse to the matrix $U_{kl} = \sum_i A_{ik} A_{il} \sigma_i^{-2}$. So, we can compute U_{kl}, invert it, and use the resulting values of $E(X_k X_l)$ to estimate $\sigma(y)$.

Hence, we arrive at the following algorithm:

Algorithm that estimates $\sigma(y)$ when errors $\sigma(x_i)$ are random.

Given:

1) An algorithm F_i that estimates x_i from z_k and t_l.

2) An algorithm F that estimates y from z_k and t_l.

3) An algorithm f that gives an estimate $\tilde{y}$ for y from $\tilde{x}_i$ by first estimating z_k and then estimating y.

4) The measured values $\tilde{x}_i, 1 \le i \le n$ and their standard deviations $\sigma(x_i)$.

5) The measured values $t_1, \ldots, t_q$, and their standard deviations $\sigma(t_l)$.

Objective: To find a standard deviation $\sigma(y)$.

Algorithm:

1) Apply f and get estimates $\tilde{z}_k$ and $\tilde{y}$.

2) Use numerical differentiation to estimate the derivatives (7) and (9).

3) Compute A_{ik}, σ_i, and U_{kl} (as above).

4) Invert the matrix U_{kl} and get $E(X_k X_l)$.

5) Substitute the resulting values of $E(X_k X_l)$ into the formula (10) and thus compute $\sigma(y)$.

Comments.

1. In this algorithm, a time-consuming "back-calculation" part is also repeated only once. Therefore, for time-consuming f, we can estimate $\tilde{y}$ and $\sigma(y)$ with practically no increase in total computation time.

2. If some parameters t_l are already known with much better precision than our measurements, then we can simplify our computations by asuming that they are absolutely precise ($\Delta t_l = 0$). In this case, it is sufficient to consider fewer than $p + q$ variables, and thus, the computation will be even smaller. Another case when we can invert matrices of sizes smaller than $(p + q) \times (p + q)$ is when some of the parameters t_l do not influence x_i at all, and are used only to compute y. For such parameters t_l, $E((\Delta t_l)^2) = \sigma(t_l)^2$, and $E(\Delta t_l \Delta z_k) = 0$ for all k.

5. Case study: pavement engineering

The problem. One of the main problems of pavement engineering is to predict the remaining lifetime y of a pavement. It is impossible to measure y directly, so instead a Falling Weight Deflectometer (FWD) is used: we drop a mass on the pavement, and measure the resulting deviations $x_1, \ldots, x_n$ in several (usually $n = 7$) locations on the surface of the pavement.

To estimate y from $\tilde{x}_i$, a 3-layer pavement model is used [9, 16]. According to this model, the pavement consists of 3 layers: 1st asphaltic/concrete, 2nd base, 3rd subgrade layer. Each layer is characterized by 3 paremeters: its thickness h_i, its Poisson ratio ν_i, and its elastic modulus E_i.

Thickness h_i does not change in time, so we can take h_i from the documents that described the design of this pavement. The values ν_i may somewhat change. However, ν_i describe the horizontal strain, and the pavement is designed for vertical loads only. So, these changes do not influence the lifetime, and we can neglect them.

So, in this problem, we have 7 parameters t_l whose values are known from the previous measurements: h_1, h_2, h_3, ν_1, ν_2, ν_3, and the weight w of the FWD mass. We have 3 parameters z_k that have to be determined from the measurements: E_1, E_2, E_3. Corresponding algorithms F_i and F are given in [9, 16] (here, F is also a two-stage algorithm: given z_k and t_l, we first compute tensile and compressive strains ε_t and ε_v, and then use these strains to predict y).

How errors were estimated before. In this problem, all the sensors are calibrated, so we can assume that the errors are random. In [12, 17], a Monte-Carlo method was used.

Computation time of the existing method. On a 386 IBM PC, forward computations (i.e., computing y and x_i from the known parameters z_k and t_l) take about 20 seconds. A program f that estimates y from $\tilde{x}_1, \ldots, \tilde{x}_n$ takes about 3 minutes to run. The Monte-Carlo method required up to 25 repetitions of this program, so its total computation time is *about 80 min.* If we take into consideration that we must measure the value of y for every mile, it is *too long.*

Our results. We applied the above algorithm to estimate $\sigma(y)$ (for details, see [3]). The entire algorithm, including both computing $\tilde{y}$ from $\tilde{x}_i$, and estimating $\sigma(y)$, ran for about 3.5 minutes. This time is close to the time (3 min) required to compute y from $\tilde{x}_i$.

Conclusion

With practically no increase in the computation time, we not only compute y, but we also estimate its precision.

References

[1] Beck, J. V. *Parameter estimation in engineering and science.* Wiley, 1977.

[2] Breipohl, A. M. *Probabilistic systems analysis.* Wiley, 1970.

[3] Chang, C.–C. *Fast algorithm that estimates the precision of indirect measurements.* Master Project, Computer Science Dept., University of Texas at El Paso, 1992.

[4] Fuller, W. A. *Measurement error models.* Wiley, 1987.

[5] *Handbook of engineering mathematics.* American Society for Metals, 1983.

[6] Kreinovich, V., Bernat, A., et al. *Parallel computers estimate errors caused by imprecise data.* Interval Computations 1 (2) (1991), pp. 31–46.

[7] Kreinovich, V. and Pavlovich, M. I. *Error estimate of the result of indirect measurements by using a calculational experiment.* Measurement Techniques **28** (3) (1985), pp. 201–205.

[8] Linnik, Yu. V. *Method of least squares and principles of the theory of observations.* Pergamon Press, 1961.

[9] Lytton, R. L. *Backcalculation of pavement layer properties.* In: Bush III, A. J. and Baladi, G. Y. (eds) "Nondestructive Testing of Pavements and Backcalculation of Moduli, ASTM STP 1026", American Society for Testing and Materials, Philadelphia, 1986, pp. 7–38.

[10] Mikhail, E. M. *Observations and least squares.* Thomas Y. Crowell, 1976.

[11] Moore, R. E. *Methods and applications of interval analysis.* SIAM, Philadelphia, 1979.

[12] Rodriguez-Gomez, J., Ferregut, C., and Nazarian, S. *Impact of variability in pavement parameters on backcalculated moduli.* University of Texas at El Paso, Dept. of Civil Engineering, Techn. Rep., 1991.

[13] Rubinstein, R. Y. *Simulation and the Monte Carlo method.* Wiley, 1981

[14] Scheaffer, R. L. and McClave, J. T. *Probability and statistics for engineers.* PWS–KENT, 1986.

[15] Sneddon, I. N. *Encyclopaedic dictionary of mathematics for engineers and applied scientists.* Pergamon Press, 1976.

[16] Uzan, J. et al. *A microcomputer based procedure for backcalculating layer moduli from FWD data.* Texas Transportation Institute Research Report 1123–1, 1988.

[17] Vennalaganti, K. M., Ferregut, C., and Nazarian, S. *Stochastic analysis of errors in remaining life due to misestimation of pavement parameters in NDT.* University fo Texas at El Paso, Civil Engineering Dept., Techn. Rep., 1992.

Received: March 10, 1993

C. FERREGUT, S. NAZARIAN, K. VENNALAGANTI
Department of Civil Engineering
University of Texas at El Paso
El Paso, TX 79968
USA

C.-C.CHANG, V. KREINOVICH
Computer Science Department
University of Texas at El Paso
El Paso, TX 79968
USA

C.-C.CHANG, CURRENT ADDRESS:
8 Kan Lo St., Taichung, Taiwan
Republic of China