

PRECISE and the reliability of Numerical Software

F. Chaitin-Chatelin* E Traviesas †

CERFACS Technical Report TR/PA/02/57

1 Introduction

It is well-known that a computation becomes unstable (and in particular computer results differ significantly from exact ones) in the neighborhood of a singularity. And a singularity is not generic : the perturbations generated by the finite-precision arithmetic are such that one always solves a regular problem. However the properties of computations are ruled by the underlying singularity, and the distance to the mathematical singularity, as viewed by the computer, is an essential parameter to interpret the computation.

We present an overview of the toolbox PRECISE (**P**recision **E**stimation and **C**ontrol **I**n **S**cientific and **E**ngineering computing) designed to assess the quality of numerical software in industrial environments, as well as for research purposes.

PRECISE is a set of tools to perform numerical experiments exploring the robustness of computational schemes. Designed for developers and users of numerical software, it provides a means for studying the reliability and stability properties of both numerical algorithms and mathematical problems. Using PRECISE, one can determine the quality of computed solutions with respect to a choice of algorithm, finite precision arithmetic, data imprecision, or any other uncertainties. When used with a backward stable algorithm, one may investigate instabilities in the physical or mathematical problem itself, particularly near singularities. Conversely, when used with well-understood mathematical problems, one can study the reliability of a numerical algorithm and its implementation.

An important aspect of PRECISE is that it is based on analysis of *computed* results, those which are actually obtained by a computer implementation of a numerical algorithm applied to a specific problem. As such, a PRECISE analysis is complementary to the standard a priori error bounds usually obtained for numerical algorithms. Moreover, since a priori error bounds are generally over-estimates of worse-case error and often assume error-free implementations, the

*Université Toulouse 1 and CERFACS (Centre Européen de Recherche et de Formation Avancée en Calcul Scientifique), 42 av. G. Coriolis, 31057 Toulouse Cedex, France, e-mail: chatelin@cerfacs.fr

†CERFACS, 42 av. G. Coriolis, 31057 Toulouse Cedex, France, e-mail: travies@cerfacs.fr

results obtained by PRECISE may be viewed as more pertinent to the problem at hand. For those who have ever obtained a numerical result and then wondered “is it correct?”, PRECISE is designed to address the inherent subtleties of this question.

PRECISE, like any tool, requires both knowledge and experience for its effective use. For this purpose, section 2 is included to provide background material on the ideas behind PRECISE. It gives an elementary introduction to the concepts necessary for experiments in numerical stability and reliability; references to the ample literature in this field are provided to allow further investigations into specific areas. For researchers already familiar to this type of numerical analysis, this section should serve as a reminder and reference during the use of the PRECISE modules.

2 Backward error analysis

The **backward error analysis**, introduced by Givens in the late fifties and developed by Wilkinson in the sixties, to analyse the reliability of algorithmic computations in a world of limited accuracy. It is the tool of choice to address the question of the quantification of the arithmetic quality of an algorithm, in order to assess the choice of this algorithm to solve a problem on a given computer. It is therefore an important method to devise **recommender systems** [25, 24, 13].

Below, all quantities overlined by a tilde represent computed versions of the exact ones.

The essence of the backward error analysis is to set the exact and the finite precision computations in a common framework by means of the following trick, that we call the **Wilkinson principle** :

Consider the computed solution \tilde{x} as the exact solution
of a nearby problem.

This almost trivial idea turns out to be much more powerful than it looks:

- i)* It allows to get rid of the details of the computer arithmetic: the errors made during the course of the computation are interpreted in terms of equivalent perturbations in the given problem, and the computed quantities are *exact* for the perturbed problem.
- ii)* One advantage is that rounding errors are put on the same footing as errors in the original data. And the effect of uncertainty in data has usually to be considered in any case.
- iii)* It enables to draw on powerful tools such as derivatives and perturbation theory.
- iv)* It allows to factor out in the error bound, the contribution of the algorithm from the contribution of the problem.

- v) Finally, it allows a great flexibility in the sensitivity analysis by providing a large choice of perturbations on the data of the problem.

Such an error analysis is referred to as backward error analysis because the errors are *reflected back* into the original problem. One essential ingredient is the **backward error** that we proceed to define.

Let $\tilde{x} = \tilde{G}(y) = \tilde{g}(z)$ be the computed solution for the problem $(P) F(x) = y$ by means of the finite algorithm G . The backward error measures the minimal distance of (P) to the set of perturbed problems that are solved exactly by \tilde{x} . Such a notion requires to specify the admissible perturbations of (P) , that is the class (τ) of admissible perturbations Δz of the data z (a subset of F, y) in a space Z and the norm $\| \cdot \|_Z$ on Z .

Definition 2.1 *The backward error at \tilde{x} associated with (τ) is defined by*

$$B(\tilde{x}) = \inf (\|\Delta z\|_Z ; \Delta z \in (\tau) \text{ and } g(z + \Delta z) = \tilde{x}).$$

This definition requires that the set $\mathcal{E} = \{\Delta z \in (\tau); g(z + \Delta z) = \tilde{x}\}$ is nonempty. The backward error at \tilde{x} gives the minimal size of the perturbation Δz of the data z when Δz varies in the set \mathcal{E} . If the class (τ) of the perturbations for Δz is a good model for the perturbations actually generated by the computer, then $B(\tilde{x})$ gives the minimal size of perturbation of the data which is equivalent to the computation in finite precision.

2.1 The consequence of limited accuracy on the data

Although the notion of backward error is mainly used by software developers and scientists to assess the quality of computer simulations, it has a broader scope.

In other words, it should not be seen as a method restricted to the analysis of the effects of round-off due to the limited precision of the computer arithmetic. In fact, it is the way to analyse the impact of a limited accuracy on the data, on a solution whether it is computed in exact arithmetic or not. Indeed it is clear in definition 2.1 that \tilde{x} does not need to be the computed solution. It can be any potential solution of a nearby problem which is indistinguishable from the original one, because of the limited accuracy on the data.

Therefore the backward error analysis is the necessary tool to assess the solution of any problem in Experimental Sciences where no data are exact (Physics, Biology, ...). It is only in Mathematics where accuracy is unlimited that alternate routes can be taken such as Computer Algebra.

2.2 Quality of reliable software

The arithmetic quality of a reliable algorithm / numerical method is related to the size of the backward error, which should be as small as possible compared to the level of accuracy on the data. The best one can do by running an algorithm on a computer, is to introduce no more uncertainty than the unavoidable one which results from introducing the data in the computer. The level of this

unavoidable uncertainty is that of machine precision $\Psi = \frac{1^+-1}{1}$. Indeed the reliability of the algorithm proves that the backward error is of order 1 in Ψ at regular points. But the algorithm is of poor quality when the constant C such that $B(\tilde{x}) \leq C\Psi$ is too large.

Definition 2.2 *The quality index of a reliable algorithm at \tilde{x} is defined by*

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

$J(\tilde{x}) \geq 1$. The best quality corresponds to $J = 1$, and a poor quality to J significantly larger than 1.

Definition 2.3 *A reliable algorithm is said to be optimal when $J \sim 1$.*

In the Numerical Software literature, the property of an algorithm that we have just defined as **optimal reliability** is referred to as **backward stability** in Numerical Software circles, after Wilkinson [27].

When the data are known with an accuracy of the order of η (i.e. $\|\Delta A\| \sim \eta \|A\|$ for instance), then the backward error should be compared to η ; this is important when η is significantly larger than machine precision. Such a situation is frequent in most applications outside mathematics: for instance, uncertainty in physical measurements lead to inaccurate data. The accuracy on the computed solution may decrease accordingly: it cannot be better than η .

The two principles of Lax [8] and Wilkinson complement each other beautifully. On the one hand, the Lax principle is used in a functional analysis framework to prove computability in finite precision. Once this first step is achieved, the Wilkinson principle is required to grade the quality of the algorithm, with machine precision or data accuracy as the gauge.

How should we stop iterative algorithms which, in exact mathematics, yield the solution after an infinite number of steps?

In this case again, the backward error is the answer: it provides a reliable stopping criterion that no forward error can match [1, 2, 8, 23].

In Numerical Software, the essence of the backward analysis is to set the exact and computed problems into the common framework of perturbation theory, in order to derive, for *regular* problems, an estimation of the error on the computed solution via the first order bound:

- (1) **Forward Error** \leq **Condition Number** \times **Backward Error**.

How accurate this error estimation is depends on the ability of the model for the perturbations (data, metrics, structure), chosen to derive the condition number and the backward error, to represent accurately the perturbations actually created by the implemented algorithm when run on a given computer [8].

3 What is PRECISE?

The toolbox PRECISE is a set of tools provided to help the user set up computer experiments to explore the impact, on the quality of convergence of numerical

methods, of finite precision as well as other types of prescribed perturbations of the data.

Because **stability** is at the heart of the phenomena under study – mathematical as well as numerical stabilities –, PRECISE allows to experiment about stability by a straightforward **randomization** of selected data, then let the computer produce a sample of perturbed solutions and associated residuals, or a sample of perturbed spectra.

The idea of using random perturbations on a selection of data, or parameters, to get information on the stability of dynamical processes is very natural and very old. It has been used extensively in Physics and technology. But it has not gained popularity in Numerical Analysis, nor in Numerical Software. However, the idea has often been recommended by the best specialists, as illustrated by the following quotation taken from [14] :

“In larger calculational problems, the relations between input data and output data are so complicated that it is difficult to directly apply the general formulas for the propagation of error. One should then investigate the sensitivity of the output data for errors in the input data by means of an *experimental perturbational calculation*: one performs the calculations many times with perturbed input data and studies the relation between the changes (perturbations) in the input data and the changes in the output data.”

This quotation serves as an excellent introduction to PRECISE, which provides an experimental environment for the engineer or the software developer to test the robustness of a numerical method or of an algorithm with respect to finite precision and data uncertainty.

It allows to perform a complete statistical backward error analysis on a numerical method or an algorithm to solve a general nonlinear problem of the form $F(x) = y$ (matrix or polynomial equation), at regular points, and in the neighborhood of algebraic singularities. It provides an estimate of the distance to the nearest singularity viewed by the computer, as well as of the order of this singularity. It can also help to perform a sensitivity analysis by means of graphical displays of samples of perturbed solutions.

PRECISE offers the following facilities [8, 23] :

- **Module 1:** a module for statistical backward error analysis providing a statistical estimation for:
 - condition numbers at regular and singular points, for the algorithm/method and the problem,
 - backward errors,
 - reliability and quality indexes,
 - distances to singularity, or dangerous borders,
 - order of Hölder-singularities.

- **Module 2:** a module for sensitivity analysis providing graphical displays of:
 - perturbed spectra,
 - spectral portraits and pseudospectra for matrices, and matrix pencils,
 - sensitivity portraits and sets of pseudozeroes for polynomials,
 - divergence portraits for iterations depending on a parameter.

The backward error is closely linked to the type of perturbations allowed on the data of the problem to be solved. Most discussions of backward error are based in terms of perturbations of the size of the machine precision Ψ , since these must always be considered with any numerical algorithm. Similarly, the backward stability of an algorithm is often expressed strictly in terms of machine precision.

However, an important idea to keep in mind for the use of PRECISE is that, if an algorithm is indeed backward stable, it will be stable for a range of perturbation sizes and not only for those close to the machine precision. Thus, one may examine numerical stability by intentionally introducing, in a controlled way, perturbations on a problem and observing the resulting outcomes. One may observe how different types of perturbations affect the results, and can lead to the discovery of sensitive or unreliable features in the solution. In cases where the perturbations reveal no instabilities, then the analysis provides the user with further evidence to the fact that the computed solution can be considered accurate, stable, and reliable, i.e. a "good" solution.

Interestingly, different types of perturbations can produce markedly different behaviour for the same algorithm applied to the same problem. It is not only a question of the size of the perturbations; the manner in which they are applied to the problem under study can affect the outcomes. The term *perturbation model* is used to describe the set of perturbations which are chosen to produce the computed results. It is important to keep in mind that the quantitative measures of reliability and sensitivity are only defined with respect to a specific perturbation model. In other words, the values for backward error and condition numbers differ according to each perturbation model. There are actually an infinite variety of perturbation models which can be applied to any problem.

Behind the use of all the PRECISE software is the use of structured perturbations on the numerical data. The notion behind a perturbation model is to mimic the effects of finite-precision arithmetic or data uncertainty to the point of producing observable, quantifiable behaviour. It is most interesting to observe the behaviour over a range of meaningful perturbation sizes and observe if the behaviour remains consistent over the entire range. If instabilities are found, then this reveals much about the numerical problem; if not, this provides strong evidence of numerical reliability.

The basic items which compose a perturbation model are the range of perturbation sizes, the type of metric for the data, the choice of data items to be perturbed, and the type of perturbations to be performed. Each of these items is explained below.

3.1 Perturbation Values

In general, a wide range of perturbation values should be used, spanning at least a few orders of magnitude. For data obtained from experimental measurements or truncated approximations, the perturbation range should include the order of the data uncertainty; one is looking to see if there is a qualitative change in the results for perturbations at this size.

For the same reason, perturbations of the same order of magnitude as the round off unit $\mathbf{u} = \frac{1}{2}\Psi$ are also frequently used. For computers implementing IEEE floating-point arithmetic, the values for the machine precision are shown in Table 1. An example range of perturbation values, therefore, might be ten values taken uniformly from the range $[2^{-54}, 2^{-10}]$. The magnitude of the perturbation values is closely related to the type of perturbations applied to the data, which are discussed below.

Single Precision	$\mathbf{u} = 2^{-24} \approx 5.96 \times 10^{-8}$
Double Precision	$\mathbf{u} = 2^{-53} \approx 1.11 \times 10^{-16}$

Table 1: Roundoff unit \mathbf{u} for IEEE standard floating-point arithmetic

3.2 Perturbation Types

For numerical linear algebra algorithms, the two most common types of perturbation models involve either normwise or componentwise perturbations. As the name implies, in a normwise model the data components of (P) are perturbed by values $\|F\|$ and $\|y\|$, which correspond to the magnitude of the data with respect to the chosen metric.

In a componentwise model the data components are perturbed according to each components individual value, $|F|$ and $|y|$, which correspond to the modulus values of the data. As an example, consider the case of studying a linear system $Ax = b$ with perturbations of size t . A normwise perturbation model would produce a set of perturbed systems of the form $(A + \Delta A)x = (b + \Delta b)$, where the perturbation matrices satisfy,

$$\begin{aligned} \|\Delta A\| &\leq t \|A\|, \\ \|\Delta b\| &\leq t \|b\|, \end{aligned}$$

While in a componentwise perturbation model, the inequalities defining the perturbations are given by

$$\begin{aligned} |\Delta A| &\leq t |A|, \\ |\Delta b| &\leq t |b|, \end{aligned}$$

where the modulus inequalities are understood to be satisfied for each component. The componentwise model is seen to preserve the same structure of the

data; in particular, for sparse matrices the same sparsity pattern is maintained for componentwise perturbations, but not for normwise perturbations.

These distinctions have originally developed because of the type of theoretical analysis performed for certain algorithms. Inspection of the above inequalities shows that a normwise analysis is based on a more global treatment of the problem than a componentwise analysis. As such, analytical error bounds obtained through a normwise analysis may be more pessimistic in some cases than those obtained from a componentwise analysis; this is the case, for example, in the solution of linear systems by Gaussian elimination.

For the purposes of PRECISE, however, these distinctions are not so critical. One powerful aspect of PRECISE is that any type of perturbation model can be used to search for meaningful results; there is no restriction in this regard. It is useful, therefore, to generalise the inequalities defining the perturbation types. The normwise perturbations must satisfy

$$\begin{aligned} \|\Delta A\| &\leq t\alpha \\ \|\Delta b\| &\leq t\beta \end{aligned}$$

for some quantities $\alpha \geq 0$ and $\beta \geq 0$ which can be adjusted according to the perturbation model. For instance, values of $\alpha \approx \|A\|$ and $\beta = 0$ indicate the same normwise perturbation on the matrix A as before, but no perturbations on the right-hand-side b . Another example might be $\alpha = 1$ and $\beta = 1$, specifying perturbations which are fixed in size for both A and b . Any combination of values for α and β gives a valid perturbation model for a study by PRECISE, except of course the combination $\alpha = 0$ and $\beta = 0$.

Similarly, the componentwise perturbations are generalised by the inequalities

$$\begin{aligned} |\Delta A| &\leq tE \\ |\Delta b| &\leq tf \end{aligned}$$

where E is any matrix of the same order as A and f is a vector of the same size as b , with both E and f having non-negative components. For problems involving sparse matrices, both E and f are usually specified to have the same structural properties as A and b , respectively.

3.3 Data Metrics

This choice pertains to the choice of norm used to measure distances in the data space. For instance, in linear algebra contexts commonly used norms are the 2-norm, the 1-norm, the ∞ -norm, or in some cases, the Frobenius norm. In Module 1, a metric is required to determine the estimators for the backward error. This choice is problem dependent, and depends also on the goals of the user. For normwise perturbation models, this choice obviously affects the actual magnitude of the perturbations on the components. To perform an error analysis, a suitable choice is often the ∞ -norm, which gives a good upper bound on error quantities and in addition is inexpensive to compute, particularly when the data is distributed in a multi-processor environment.

3.4 Data to be perturbed

One has the flexibility to perturb all or just a subset of the data components in the problem. For example, when studying a linear system $Ax = b$, one can perturb only the matrix A , only the right-hand-side b , or both A and b . One might also choose to perturb only a few elements of A , for instance, those which correspond to certain critical unknown variables. One usually starts with a simple approach, such as perturbing all data elements, and then trying other combinations for comparison.

3.5 Choosing a Perturbation Model

While there are an infinite number of possibilities for choosing a perturbation model, there are some clear guidelines to follow to make the choices quite straight-forward. Most often, when performing a stability analysis, a series of perturbation models will be used to gain further and further insight into the numerical behaviour of the problem and its solution. Therefore, one may start with almost any model, and then refine the choices according to the phenomena which are observed. The choices are always problem dependent, but the following general principles can be used to guide the analysis.

Initially, one usually starts with a wide range of perturbation sizes which includes the machine precision or any level of data uncertainty. For example, if the numerical procedure calls for a parameter to define the error tolerance of the approximated solution, then the perturbation range should include this value. A special case of this situation is the specification of the stopping criteria for iterative solvers; for example, if the iterations are stopped using a criteria of 10^{-k} , then the perturbation interval should bracket this value. While the range of perturbation values can initially be quite large, the actual number of perturbation sizes taken from the range does not have to be large also. Five to ten perturbation sizes taken from the range $[\xi^{-k_{\min}}, \xi^{-k_{\max}}]$ should suffice for initial tests. As values for the stability indicators are observed, one may increase the number of perturbations taken from the range, and also adjust the values of k_{\min} and k_{\max} accordingly. Note that the user is free to specify any value for ξ , but a value of $\xi = 2$ allows a fine sampling of interval range.

The choices of perturbation types, data metrics, and the data to be perturbed are all very problem dependent. If the numerical algorithm under study is known to exhibit certain behaviour for a particular type of perturbation, then this most certainly should be used in a stability analysis. On the other hand, if no theoretical behaviour is predicted, then the user is free to make these choices based on other criteria. Notably, the manner in which the numerical procedure is implemented can be used to guide the choice for an initial perturbation model. In other words, the model which is most conveniently implemented may be used first. See also section 3.7 for a discussion on implementation issues concerning perturbation models.

Finally, after an initial perturbation model has been selected and the results interpreted, the choice of subsequent perturbation models will depend on the observed results. There are in general three categories of behaviour which might

be observed. If the results show no evidence of numerical instability, then this provides corroborative evidence of the reliability of the numerical procedure. In this case, one would like to apply more rigorous tests to the problem in the hope that the same positive results can be obtained. The more tests that are applied with positive outcomes, the stronger the validation of the numerical procedure. For this situation, the user is free to change the perturbation model in any number of ways, for instance by changing the perturbation sizes, the perturbation type, or the data metric. The process can continue in this way until the user is confident in the results.

The second category of results involves the situation when instabilities are observed. Thus, the subsequent perturbation models should be designed to investigate this condition. If instabilities are observed, further insight can be gained by using the same perturbation sizes and changing the type of perturbation or by limiting the data which is being perturbed. In such cases, it is not uncommon for different perturbation models to produce remarkably different behaviour in the numerical problem; the results are not contradictory, but actually provide a more complete characterisation of the numerical properties of the problem under study.

It can happen that the indicators may be inconclusive for some perturbation values. If the indicators show inconclusive results, this can either be the result of an improper perturbation model or an actual problem in the numerical procedure. In this case, the range of perturbation values should be changed to determine if the indicators reveal different results. Different perturbation types or data metrics may also be used. If a number of perturbation models are attempted without conclusive results, then most likely some anomaly in the numerical procedure is present. For example, a numerical procedure which returns a constant error value for a certain range of perturbation values would exhibit this behaviour. Thus, in such cases, the numerical procedure requires a more thorough inspection to determine the cause.

3.6 Implementation Issues

A backward error analysis requires a small amount of programming effort.

- The first task is to implement the perturbation model, including the iteration to produce samples of perturbed solutions for each perturbation size.
- A second task involves processing the samples to calculate the statistical indicators. The processing may be done simultaneously while the data is generated, or the samples may be written to a file and the indicators calculated in a separate phase.
- A final task is to visualise the stability indicators once they have been calculated. This task depends on the type of graphical software available to the user.

These issues are examined in the following paragraphs. For examples of completed implementations of backward error analysis routines using popular software libraries, see [8, 23].

3.7 Implementing a Perturbation Model

A general discussion of perturbation models is given in Section 3. Here, some practical concerns for implementing a perturbation model are discussed. One should also be aware of these issues when deciding what perturbation model(s) to apply for a study with PRECISE; the choice of perturbation model may depend on how it is to be implemented.

The main idea is to use perturbation sizes t taken from a range of values $[\xi^{-k_{\max}}, \xi^{-k_{\min}}]$. For each perturbation size, a number of samples of perturbed solutions shall be produced. A starting point is how to specify the perturbation values. Since in general a wide range of values is required which spans orders of magnitude, a simple method is to specify the lower and upper exponents of the range. That is, using some base value such as $\xi = 2$ for instance, one need only specify the range $[\xi^{-k_{\max}}, \xi^{-k_{\min}}]$ by providing k_{\min} and k_{\max} . Any base value ξ can be used. However, the particular choices of $\xi = 10$ or $\xi = 2$ are usually the most convenient; the former value allows perturbations of corresponding digits in the data components, while the latter value corresponds to bit-wise perturbations of the data. To store the perturbation values in an array, a counter k can be incremented from k_{\min} and k_{\max} to produce a uniform sample of perturbation values t . In a distributed parallel environment, each processor can perform this operation without communication, as long as k_{\min} and k_{\max} are constants known to all processors. Note that the particular order in which the perturbation sizes are used to produce indicators is usually not important; they can be increasing or decreasing in size during the indicator calculations as long as the correct order is presented during their visualisation.

A number of samples will be generated for each perturbation size t . The number of samples does not need to be large, usually between 5 to 15 samples gives very satisfactory results [8]. The number of samples should remain constant for all the different perturbation values taken from the perturbation range.

A random number generator is also required. All the routines in PRECISE use the routine named RANDOM, which can be modified according to system requirements (see Section 3.3 in [23]). If perturbations are to be produced on distributed data in a parallel environment, care should be taken to ensure the sequences of pseudorandom numbers on each processor are independent; this can be done by using different seeds on each processor.

Another issue concerns perturbations of real or complex data. When randomly perturbing a real data value x by a perturbation size t , the result is either $x+t$ and $x-t$, each having equal probability. However, when randomly perturbing a complex data value, care should be taken since random number generators usually produce only real random (or more precisely, pseudorandom) numbers. To perturb a complex data value z by a perturbation size t , let r represent a random number between 0 and 1, then a valid perturbation is given by $z + w$, where $w = te^{i2\pi r}$.

In some cases, the perturbations may need to be adjusted to suit a particular data metric. For example, when performing a normwise analysis on matrices,

care should be taken to ensure the inequalities defining normwise perturbations are satisfied. The type of norm used may require additional scaling of the perturbation values. In particular, using the $\| \cdot \|_\infty$ matrix norm, the perturbation on each component should be scaled according to the number of non-zero row elements. This is particularly important for large, sparse matrices, where one usually only chooses to perturb the non-zero elements since algorithms for these problems are designed specifically to manipulate only these elements.

A diagram of the general procedure for producing the sample data is given in Table 2. First, the numerical problem under study is solved without introduc-

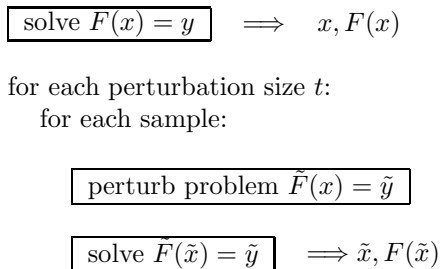


Table 2: General scheme for generating data for a backward error analysis. The data is indicated by arrows (\implies). Note that after each perturbed solution \tilde{x} is obtained, the unperturbed mapping F (not \tilde{F}) is used to generate the image $F(\tilde{x})$

ing user-defined perturbations. Following this, an iteration is required, where a number of samples of perturbed problems are solved for each perturbation size. For each perturbed solution \tilde{x} obtained, both \tilde{x} and the value obtained by the reverse mapping $F(\tilde{x})$ are required for the analysis. Note that in general $F(\tilde{x})$ differs from $\tilde{F}(\tilde{x})$, and only the former quantity is needed. For example, if the problem under study is a linear system $Ax = b$ and the perturbed problem $A\tilde{x} = b$ is solved, then the quantities used for the analysis are \tilde{x} and $A\tilde{x}$.

For most applications, it should be quite straight-forward to place the numerical solver in an iterative procedure as shown in Figure 2. In general, the iteration requires a separate copy of all the input data for the problem to be saved so that it will be available for each subsequent perturbed problem. If this is not feasible due to memory constraints, the initial data for the problem may be re-loaded from disk files for each perturbation. While this can be time-consuming, the issue of efficiency is not the primary concern during a backward error analysis. If the numerical solver has been designed to work in a parallel environment, one should ensure that the processors are synchronised throughout the iteration.

The example software discussed in [23] can be used to generate perturbations for most types of problems. In those examples, routines to produce perturbations on vectors can be used for matrices as well. Modifications to these routines to produce more complicated perturbation models can be easily performed. In addition, other perturbation techniques are possible for special cases. For in-

stance, since iterative algorithms for large, sparse matrices usually access the matrix data only through matrix-vector multiplications, it may be worthwhile to perturb only the vector resulting from the matrix-vector product. As explained above, any approach to producing perturbations can produce valid results in the context of PRECISE.

4 Industrial use of PRECISE

PRECISE has been intensively used, since 1988, in several industrial environments (IBM-France, Thomson-CSF and CERFACS) to test various *laws of computation* that emerge from invariant patterns of behavior for computations in finite precision (see [8]) It has also been used, more classically, to assess the numerical quality of computations in industrial problems such as

- the flutter phenomenon for Aerospatiale [5, 19],
- an aeroelasticity problem for ONERA (Division Hélicoptères) [3, 4],
- electromagnetic guided waves for Thomson-CSF [4],
- the reliability of an orbitography software for CNES [11, 9, 16, 20, 21],
- fluid dynamics and electromagnetism at CERFACS [15, 18, 26],
- ambiguity resolution in GPS (Jason project) for CNES [7],
- astrophysics for Observatoire Midi-Pyrénées and Politechnico Milano.

The PRECISE code was a vital part of the HPCN (High Performance Computing and Networking) European project PINEAPL (1996-98) to produce a general purpose library of parallel numerical software suitable for a wide range of computationally intensive industrial applications and to port several application codes which use this library to parallel computers. The industrial consortium lead by NAG included British Aerospace, CERFACS, LCR Thomson-CSF, CPS (Napoli, Italy), the Danish Hydraulic Institute (Denmark), IBM SEMEA, the University of Manchester (UK), Math-Tech, and Piaggio (Italy).

The toolbox PRECISE for MATLAB given in Annex of [8] was translated into Fortran to allow large and realistic problems to be handled and was used to **test each item of numerical software produced during the project.**

The Fortran toolbox is now available as freeware from CERFACS

<http://www.cerfacs.fr/algos/Softs/PRECISE/precise.html> [23].

Since July 2000, 86 Matlab and Fortran PRECISE codes have been downloaded from this web site by users from Argentina, China, England, France, Greece, Ireland, Marocco, Russia, Spain and U.S.A.

5 PRECISE in academic research

In parallel to its use in various industrial environments PRECISE has been used to test many conjectures about computer simulations in the neighbourhood of singularities. Such conjectures range from the influence of nonnormality on iterative processes [10, 12] and on eigenvalues [3, 6, 5, 19] to the assessment of computation near algebraic singularities. The computation of a Jordan form has been a point of special focus [22, 12]

We conclude by stating that PRECISE has proved an extremely versatile tool to test algorithmic behavior on the computer as illustrated by the following example. The quantity $\|A^{-1}\|_2$, where $\|\cdot\|_2$ is the spectral norm of a very large matrix A can be computed using Lanczos algorithm on A^*A , with `invert`. Such an algorithm requires the solution of linear systems of the kind $A^*Ax = b$.

Extensive experimentations with PRECISE lead to the discovery of the theoretical formulae for the condition number and the backward error for such linear systems where only A is perturbed. This study is of importance to design a reliable code for computing the spectral portrait of a matrix [17, 23]. **So PRECISE helped design itself.**

For more, the interested reader is referred to the Qualitative Computing Group web pages at

<http://www.cerfacs.fr/algor/qualitative/Qualitative.html>

Conclusion

The toolbox PRECISE (**P**recision **E**stimation and **C**ontrol **I**n **S**cientific and **E**ngineering computing) is not intended to be yet another software for automatic control of round-off error propagation. It is as much a matter of personal taste as of performance that should guide the user's choice amongst the available methods and software for automatic control of accuracy.

We view PRECISE as a help to investigate difficult cases, such as computations in the neighborhood of a singularity, or computations in the presence of high nonnormality, to get better insight on the underlying mathematical instability.

In particular PRECISE is a big help in the many cases where there do not exist yet any formulae for backward error or condition number. The better understanding of the problem provided by PRECISE allows in turn a better use of current software for error control. In other words, PRECISE helps defining the limits of recommendation for certain recommender systems.

References

- [1] M. Arioli, I.S. Duff, and D. Ruiz. Stopping criteria for iterative solvers. *SIAM J. Matrix Anal. Appl.*, 13:138–144, January 1992.

- [2] R. Barrett, M. Berry, T. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst. *Templates for the solution of linear systems : building blocks for iterative methods*. SIAM, Philadelphia, 1994.
- [3] M. Bennani, T. Braconnier, and J.-C. Dunyach. Solving large-scale non-normal eigenproblems in the aeronautical industry using parallel BLAS. In W. Gentzsch and U. Harms, editors, *High-Performance Computing and Networking*, volume 796, pages 72–77. Springer-Verlag, 1994.
- [4] T. Braconnier. *Sur le calcul de valeurs propres en précision finie*. Ph.D. dissertation, Université H. Poincaré, Nancy, May 1994. TH/PA/94/24, CERFACS.
- [5] T. Braconnier, F. Chatelin, and J.-C. Dunyach. Highly nonnormal eigenvalue problems in the aeronautical industry. *Japan J. Ind. Appl. Math.*, 12:123–136, 1995.
- [6] F. Chaitin-Chatelin. Is nonnormality a serious computational difficulty in practice ? In R. Boisvert, editor, *The quality of Numerical Software*, pages 300–314. Chapman and Hall, 1997.
- [7] F. Chaitin-Chatelin, S. Dallakyan, and V. Frayssé. An overview of Carrier Phase Differential GPS. Contract Report IR/PA/99/50, CERFACS, 1999.
- [8] F. Chaitin-Chatelin and V. Frayssé. *Lectures on Finite Precision Computations*. SIAM, Philadelphia, 1996.
- [9] F. Chaitin-Chatelin, V. Frayssé, and S. Gratton. Fiabilité des calculs en précision finie : principes et mise en œuvre. Contract Report FR/PA/97/55, CERFACS, 1997.
- [10] F. Chaitin-Chatelin and S. Gratton. Convergence of successive iteration methods in finite precision under high nonnormality. *BIT*, 36:455–469, 1996.
- [11] F. Chaitin-Chatelin and S. Gratton. Etude de la non convergence de Gauss-Newton. Contract Report IR/PA/97/14, CERFACS, 1997.
- [12] F. Chaitin-Chatelin, A. Harrabi, and A. Ilahi. About Hölder condition numbers and the stratification diagram for defective eigenvalues. In Elsevier, editor, *Mathematics and Computers in Simulation*, volume 54, pages 397–402, 2000.
- [13] F. Chaitin-Chatelin and E. Traviesas. PRECISE, a toolbox for assessing the quality of numerical methods and software. Technical Report TR/PA/00/12, CERFACS, August 21-25,2000. Proceedings of the 16th IMACS World Congress, Lausanne, Switzerland.
- [14] G. Dahlquist and Å. Björck. *Numerical methods*. Prentice-Hall, Englewood Cliffs, NJ, 1980.

- [15] V. Frayssé, L. Giraud, and V. Toumazou. Parallel computation of spectral portraits on the Meiko CS2. In H. Liddell, A. Colbrook, B. Hertzberger, and P. Sloot, editors, *High-Performance Computing and Networking*, volume 1067, pages 312–318. Springer-Verlag, 1996.
- [16] V. Frayssé and S. Gratton. Moindres carrés pour l’orbitographie - Etude de stabilité - Partie II. Contract Report FR/PA/95/28, CERFACS, 1995.
- [17] V. Frayssé, S. Gratton, and V. Toumazou. Structured backward error and condition number for linear systems of the type $A^*Ax = b$. *BIT*, 40:74–83, 2000.
- [18] V. Frayssé, M. Gueury, F. Nicoud, and V. Toumazou. Spectral portraits for matrix pencils: a physical application. Technical Report TR/PA/96/19, CERFACS, 1996.
- [19] S. Godet-Thobie. *Eigenvalues of large highly nonnormal matrices (Valeurs propres de matrices hautement non normales en grande dimension)*. Ph.D. dissertation, Université Paris IX Dauphine, December 1992.
- [20] S. Gratton. On the condition number of linear least squares problems in a weighted Frobenius norm. *BIT*, 36:523–530, 1996.
- [21] S. Gratton. *Outils théoriques d’analyse du calcul à précision finie*. Ph.D. dissertation, Institut National Polytechnique de Toulouse, June 1998. TH/PA/98/30, CERFACS.
- [22] A. Ilahi. *Validation du calcul sur ordinateur: application de la théorie des singularités algébriques*. Ph.D. dissertation, Université Toulouse I, June 1998. TH/PA/98/31, CERFACS.
- [23] R. A. McCoy and V. Toumazou. PRECISE User’s Guide - Version 1.0. Tech. Rep. TR/PA/97/38, CERFACS, 1997.
- [24] N. Ramakrishnan and C.J. Ribbens. Mining and visualizing recommender system for one-dimensional numerical quadrature. *ACM Transactions on Mathematical Software*, 2000. To appear.
- [25] N. Ramakrishnan, J.R. Rice, and E.N. Houstis. GAUSS : An online algorithm recommender system for one-dimensional numerical quadrature. *ACM Transactions on Mathematical Software*, 2000. To appear.
- [26] V. Toumazou. *Portraits spectraux de matrices: un outil d’analyse de la stabilité*. Ph.D. dissertation, Université H. Poincaré, Nancy, 1996. TH/PA/96/46, CERFACS.
- [27] J. H. Wilkinson. Error analysis of floating point computation. *Numer. Math.*, 2:219–340, 1960.