Checking Approximate Computations over the Reals

Sigal Ar[*] Manuel Blum[†] Bruno Codenotti[‡]

Pete Gemmell[§]

TR-92-030

May 1992

Abstract

This paper provides the first systematic investigation of checking approximate numerical computations, over subsets of the reals. In most cases, approximate checking is more challenging than exact checking. Problem conditioning, i.e., the measure of sensitivity of the output to slight changes in the input, and the presence of approximation parameters foil the direct transformation of many exact checkers to the approximate setting. We can extend exact checkers only if they have a very smooth dependence on the sensitivity of the problem. Furthermore, approximate checking over the reals is complicated by the lack of nice finite field properties such as the existence of a samplable distribution which is invariant under addition or multiplication by a scalar. We overcome the above problems by using such techniques as testing and checking over similar but distinct distributions, using functions' random and downward self-reducibility properties, and taking advantage of the small variance of the sum of independent identically distributed random variables.

[*]Department of Computer Science, Princeton University, Princeton, NJ 08544-2087. Supported by NSF PYI grant CCR-9057486 and a grant from MITL.

[†]Computer Science Division, UC Berkeley, Berkeley, CA 94720, and International Computer Science Institute, Berkeley CA 94704. Supported by NSF grant CCR88-13632.

[‡]International Computer Science Institute, Berkeley, CA 94704, and IEI-CNR, Pisa (Italy). Partially supported by the "Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo". Subproject 2. e-mail: brunoc@icsi.berkeley.edu

[§]Computer Science Division, UC Berkeley, Berkeley, CA 94720. Supported by GTE, Schlumberger Fellowships, and by NSF grant CCR88-13632.

We provide checkers for a variety of approximate computations, including matrix multiplication, linear system solution, matrix inversion, and computation of the determinant. We also present an approximate version of Beigel's trick and extend the approximate linear self tester/corrector of [8] to more general computations including trigonometric functions.

Our techniques are very general and we believe that they will have applications to a broader class of problems such as the solution of Riccati and other matrix equations, computation of matrix functions, and several other polynomial and non-linear computations.

We conjecture that the approximation parameters in our checkers are optimal. The intuition behind this is that their ratio is proportional to the reciprocal of the distance between the given instance and a singularity.

1 Introduction

The notion of program checking is due to Blum [2] and has been developed by [14, 3, 12]. In this paper we propose a framework within which to use the ideas of program checking in the environment of numerical algorithms, where the computation is approximate, due for example to the propagation of roundoff errors.

When dealing with approximations some certainties are lost. A mathematical structure is transformed into a more complicated one with weaker operators. We found evidence of the hardness of approximations when looking at the task of designing approximate result checkers. The goal of approximate checking is to distinguish between programs which produce small errors and those that produce large errors. In particular, one might distinguish small errors, caused by roundoff, from larger errors, due to bugs or catastrophic propagation of roundoff.

Checking computations which do not produce exact results is a process which can be very different from checking exact computations. There are at least two reasons. First, approximate checking introduces a grey zone: there are two approximation parameters, ϵ_1 and ϵ_2. We must almost surely pass all programs which are within ϵ_1 of the correct answer for all inputs, and we must almost surely fail programs not within ϵ_2 of the correct answer on the given instance. The task is to make ϵ_1 and ϵ_2 as close as possible.

Second, there is the issue of problem conditioning, i.e. the measure of how sensitive the output is to slight changes in the input. A test whose exact fulfillment is necessary and sufficient for an exact solution might not transfer nicely to the approximate case. By this we mean that the test's approximate fulfillment might not be necessary or might not be sufficient for an approximate solution. One case where we encountered this problem was in designing a checker for linear system solution. If x is the exact solution to a linear system $Ax = b$ and $\hat{x}$ is an estimate for x, then we do not necessarily have that $||A(\hat{x} - x)||$ is small iff $||\hat{x} - x||$ is small.

Our work provides a substantial improvement over the state of the art of approximate program checking. Previous results extended existing checkers for exact computations to the approximate setting [14, 8]. They all took advantage of properties of finite groups. These previous approximate checkers were available only for generalized univariate linear functions and polynomials over finite groups and three elementary functions on floating point domains (exponential, logarithm, and quotient).

We provide approximate checkers for a variety of numerical problems in linear algebra. By necessity most of these checkers are substantially different from the checkers for the exact versions of the same problems.

- **Matrix multiplication.** We modify Freivald's exact checker [7]. We use vectors with entries from $\{-1, 1\}$ so as to minimize the distance between the two approximation parameters.

- **Linear system solution.** After we show that the naive approach of plugging the solution back into the system does not work, we provide a checker which is based on contructing instances with known solution over one randomized distribution, testing the program over this distribution, and than correcting over a randomized distribution which is very close to the one we tested over.

- **Matrix inversion.** We take advantage of the fact that a nonsingular matrix and its inverse satisfy a simple matrix equation. This allows us to use modified Freivald's checker for matrix multiplication.
- **Computation of the determinant.** We provide two checkers. One has general application but relatively loose bounds. The other achieves better bounds, but applies to matrices with special properties.

In addition, we provide checkers for *generalized homomorphic functions*, and some trigonometric functions.

For lack of space, we did not put emphasis on the presentation of several other important results that will appear in greater detail only in the full paper. These include an approximate version of Beigel's trick [1], checkers for approximate parallel programs and checkers for special cases such as linear algebra computations with Toeplitz, triangular, and band matrices.

In the course of writing this paper, we noted that the checkers for numerical problems fall into three different categories:

1. We can take advantage of functions which are downward self-reducible by following one line of the recursion toward a small instance and checking the program on that instance. This is the case with the first determinant checker.
2. For functions f which are random self-reducible to problems of the same input size and for which we are able to compute $f(R)$ easily for random instances R, we get both approximate self-testers and correctors. This is the case with the linear system solution checker.
3. For some functions, f, such as matrix multiplication, we are able to take advantage of a necessary and sufficient property of f that can be checked in low time complexity.

The rest of this paper is organized as follows. In section 2 we give some definitions; in section 3 we describe the model for approximate checking. In section 4 we describe the approximate checkers; in section 5 we give some extensions and further results. Section 6 contains concluding remarks, and the appendix contains some proofs from section 4.

2 Definitions

Definition 1 *(matrix norm) For any given norm $\|\cdot\|$ on vectors, its associated* (operator) norm on matrices, *for any matrix A, is defined as*

$$\|A\| = \max_{\mathbf{x}\neq 0} \frac{\|A\mathbf{x}\|}{\|\mathbf{x}\|} = \max_{\|\mathbf{x}\|=1} \|A\mathbf{x}\|. \tag{1}$$

In this paper, we will make use of the following norms:

1. The *infinity* vector norm, $\|x\|_\infty = max_i |x_i|$;
2. The *infinity* matrix norm, $\|A\|_\infty = max_i \sum_j |a_{ij}|$.

The notation $\|\cdot\|$ stands for any norm.

Definition 2 *(condition number) The function $\mu(A)$, defined by*

$$\mu(A) = \begin{cases} \|A\| \cdot \|A^{-1}\| & \text{if } A \text{ is nonsingular,} \\ \infty & \text{otherwise,} \end{cases}$$

is called the condition number of A.

Definition 3 *("little-oh" property) The checkers must have a running time which is asymptotically less than the running time of the program being checked, when each call to the program counts as one step in the checker's computation.*

Definition 4 *(approximation) We say that a program $P(\delta, \epsilon)$-***approximates** *the function f, if $\|P(x) - f(x)\| \leq \epsilon$ for all but a δ fraction of the inputs.*

Definition 5 *(approximate result checker) Let $0 \leq \epsilon_1 \leq \epsilon_2$. An (ϵ_1, ϵ_2)-***approximate result checker** *for a function f, under a given norm $\|\cdot\|$ (or $|\cdot|$ for real numbers), is a probabilistic oracle program C_f, which is used to check the quality of the approximation of any program that claims to approximate f. This is done as follows. On a given input x, Program P, and security parameter β, C_f^P must behave as follows:*

1. *If $P(x)$ is not within ϵ_2 of $f(x)$, in the sense that $\|P(x) - f(x)\| > \epsilon_2$ (or $|P(x) - f(x)| > \epsilon_2$), then $C_f^P(x)$ outputs "FAIL P", with probability at least $1 - \beta$.*
2. *If $\forall y$, $P(y)$ is within ϵ_1 of $f(y)$, in the sense that $\|P(y) - f(y)\| \leq \epsilon_1$ (or $|P(y) - f(y)| \leq \epsilon_1$), then $C_f^P(x)$ outputs "PASS P(x)", with probability at least $1 - \beta$.*

Definition 6 *(approximate self-tester) Let $0 \leq \delta_1 \leq \delta_2 < 1$. Let $0 \leq \epsilon_1 \leq \epsilon_2$. A $(\delta_1, \delta_2, \epsilon_1, \epsilon_2)$-***approximate self-tester** *for the function f, is a probabilistic oracle program T_f so that on a given program P, and security parameter β, T_f^P must behave as follows:*

1. *If P (δ_1, ϵ_1)-approximates f, then T_f^P outputs "PASS", with probability at least $1 - \beta$.*
2. *If P does not (δ_2, ϵ_2)-approximates f, then T_f^P outputs "FAIL", with probability at least $1 - \beta$.*

Definition 7 *(approximate self-corrector) Let $0 \leq \delta < 1$. Let $0 \leq \epsilon_1 \leq \epsilon_2$. A $(\delta, \epsilon_1, \epsilon_2)$-***approximate self-corrector** *for the function f, is a probabilistic oracle program C_f that has the following property on a given input x, program P, and security parameter β. If P (δ, ϵ_1)-approximates f, then $C_f^P(x)$ is within ϵ_2 of $f(x)$, with probability at least $1 - \beta$.*

Note that it is easy to obtain an approximate result checker from an approximate self-tester combined with an approximate self-corrector.

Notation. We denote by $x \in_U S$ the fact that x is an element of S, uniformly chosen.

3 The Model

In this section we will define what we mean by computation over the reals, and by checking this type of computation. We assume that the inputs to the program to be checked are a set of real numbers. The output is one or more real number. This, of course, includes the special case where the program's computation is over a small subset of the reals, e.g. the rationals or a set of fixed or floating point numbers. In all previous work on checkers, it is assumed with loss of generality that the checker is faultless. We continue using previous assumptions here, further assuming (with "some" loss of generality) that the checker's computations have negligibly small error compared to the program being checked. This is a reasonable assumption to make because: (1) the checker does less total computation than does the program (the "little-oh" property); (2) our checkers are very simple and built up by using very basic numerical primitives so that we can use extended precision arithmetic to make the checker's computation exact or extremely accurate; (3) all checkers which we present would still be valid if the errors which they introduce were small relative to that of the program.

For some of the linear algebra checkers, we ask the checker to sample from the uniform distribution on a bounded set of vectors. In practice, the checker needs only to sample from a set of vectors that can be written to some sufficiently good finite precision.

Note that, in general, the program error can come from one of three sources: round-off, bugs, or approximation (such as that associated with iterative methods, even when using exact arithmetic).

4 Checkers for numerical computations

4.1 Matrix Multiplication

The importance of checking matrix multiplication algorithms follows from the existence of asymptotically fast algorithms solving the problem in a rather complicated way (see, e.g., [6]). Here we present a result checker for approximate matrix multiplication based on Freivald's idea [7].

Recall that Freivald's checker is based on the following property. If A, B and C are $n \times n$ matrices, with $(AB)_{ij} \neq C_{ij}$, and r is an n-vector with $0, 1$ entries so that $ABr = Cr$, then if $\bar{r}$ is the vector obtained from r by toggling the j-th bit, we have that $AB\bar{r} \neq C\bar{r}$. We assume that the input is two $n \times n$ real matrices A and B and the output $P(A, B)$ of a program P claiming to approximate AB. The following is the checker we propose:

```
Program Matrix Multiplication Approximate Check($A, B, P(A,B), \epsilon, \beta$)
    Repeat $log(\frac{1}{\beta})$ times
        Choose $v \in_U \{-1,1\}^n$
        Check $||P(A,B)v - A(Bv)||_\infty \le \frac{\epsilon}{4}$
    If any of the checks fail then output FAIL $P$; otherwise output PASS $P(A,B)$
```

Theorem 1 *The above protocol is an $(\frac{\epsilon}{4}, \sqrt{n}\epsilon)$-approximate checker for $n \times n$ matrix multiplication, under the $\|\cdot\|_\infty$ norm.*

Proof. We need to prove the following:
1. If $||P(A,B) - AB||_\infty > \sqrt{n}\epsilon$, then $Pr[P \text{ fails }] > 1 - \beta$.
2. If $||P(A,B) - AB||_\infty \le \frac{\epsilon}{4}$, then $Pr[P \text{ passes }] > 1 - \beta$.
1. Let $e_{i,j} = (P(A,B) - AB)_{i,j}$. There exists specific row i_0 such that $\sum_{j=1}^{n} |e_{i_0 j}| > \sqrt{n}\epsilon$. In the worst case, $\forall j \in \{1 \ldots n\}, |e_{i_0,j}| \ge \frac{\epsilon}{\sqrt{n}}$. Then, using Stirling's approximation, $Pr_{v \in_U \{-1,1\}^n}[|\sum_{j=1}^{n} v_j e_{i_0 j}| > \frac{1}{4}\sqrt{n}\frac{\epsilon}{\sqrt{n}} = \frac{\epsilon}{4}] > \frac{1}{2}$. The probability that P fails is more than $1 - \frac{1}{2}^{\log_2 \frac{1}{\beta}} = 1 - \beta$.
2. We have $\sum_{j=1}^{n} |e_{i_0 j}| \le \frac{\epsilon}{4}$. Then, with probability 1, $|\sum_{j=1}^{n} v_j e_{i_0 j}| \le \frac{\epsilon}{4}$.

□

4.2 Linear System Solution

Here we consider checking programs $P(= P_A)$ which claim to solve $n \times n$ linear systems of the form $Ax = b$ where A is a real $n \times n$ nonsingular matrix and b is a real n-vector. We denote by $P(b) = P_A(b)$ the output of the program on matrix A and vector b. The checker is also provided with a positive number γ such that $\forall x, ||Ax|| \ge \gamma||x||$. This number is related to the minimum *singular value* of A and it is a measure of how close A is to a singularity (see [9]). We can either assume that γ is given, or we may assume to have a library (see [4]) containing a tested program for the *singular value decomposition* of A which can be used to check γ.* We note here that simply computing $||A(P(b)) - b||$ is not a satisfactory check because it may yield poor approximation parameters. This is due to the fact that the value of $||A(P(b)) - b||$ can be small and at the same time the value of $\|x - P(b)\|$ can be as large as $\|A^{-1}\| \cdot \|A(P(b)) - b\|$.

Let $\mathcal{D}$ be the n dimensional cube of vectors, $x =< x_1, \ldots, x_n >$ with $-\frac{||b||_\infty 10n}{\gamma} < x_i < \frac{||b||_\infty 10n}{\gamma} \forall i$. Define $A\mathcal{D} = \{Ax : x \in \mathcal{D}\}$. Our approximate checker requires that the uniform distributions on $A\mathcal{D}$, on which we test, and on $b - A\mathcal{D}$, on which we check, differ only very slightly. This is why we define $\mathcal{D}$ so that the sides of the box $A\mathcal{D}$ will be large relative to b.

*A development of the library notion will be provided in the full paper.

```
Program Linear System Solution Approximate Check($P, A, b, \gamma, P(b), \epsilon, \beta$)
Approximate Self-Test
  Repeat $\log_{\frac{3}{2}}(\frac{2}{\beta})$ times
    choose $y \in_U \mathcal{D}$ and test $||P(Ay) - y||_\infty \leq 2\epsilon$
  If any of the tests fail, then output FAIL $P$
Approximate Self-Check
  Repeat $\log_2(\frac{2}{\beta})$ times
    choose $y \in_U \mathcal{D}$ and check $||P(b) - (y + P(b - Ay))||_\infty \leq 2\epsilon$
  If any of the checks fail then output FAIL $P$; otherwise output PASS $P(b)$
```

Note that the above self-check can easily be converted into a self-corrector if the approximate solution is computed as the componentwise median of $y + P(b - Ay)$.

Theorem 2 *The above protocol is an $(\epsilon, 4\epsilon)$-approximate checker for linear system solution, under the $\|\cdot\|_\infty$ norm.*

Proof. We need to prove the following:
1. If $\forall\, v \in A\mathcal{D} \cup b - A\mathcal{D}$, $||P(v) - A^{-1}v||_\infty \leq \epsilon$ then $Pr[P \text{ passes }] > 1 - \beta$.
2. If $||P(b) - A^{-1}b||_\infty > 4\epsilon$ then $Pr[P \text{ passes }] \leq \beta$.
1. First we show that $b \in A\mathcal{D}$. Note that $||A^{-1}||_\infty \leq \frac{1}{\gamma}$ because $\forall x, \gamma||x|| \leq ||Ax|| \Rightarrow \forall y, ||A^{-1}y|| \leq \frac{1}{\gamma}||y||$. $||A^{-1}b||_\infty \leq \frac{1}{\gamma}||b||_\infty < \frac{10n||b||_\infty}{\gamma} \Rightarrow A^{-1}b \in \mathcal{D} \Rightarrow b \in A\mathcal{D}$. Therefore $\forall v \in A\mathcal{D}, ||P(v) - A^{-1}v||_\infty \leq \epsilon$ implies $||P(b) - A^{-1}b||_\infty \leq \epsilon$. P with b will pass both the approximate self-test and the approximate self-check with probability 1.
2. Assume that $||P(b) - A^{-1}b||_\infty > 4\epsilon$. If $Pr_{v \in_U A\mathcal{D}}[||P(v) - A^{-1}v||_\infty \leq 2\epsilon] < 2/3$ then $Pr[P \text{ passes the approximate self-test }] < \frac{2}{3}^{\log_{\frac{3}{2}}\frac{2}{\beta}} = \frac{\beta}{2}$. From now we assume that $Pr_{v \in_U A\mathcal{D}}[||P(v) - A^{-1}v||_\infty \leq 2\epsilon] \geq 2/3$. Since $||b||_\infty$ is small relative to the length of the sides of $A\mathcal{D}$, which is an n-dimensional box centered at the origin in $\mathbf{R}^n$, we know that the uniform distributions on $A\mathcal{D}$ and $b - A\mathcal{D}$ differ only very slightly. Specifically, the two sets $A\mathcal{D}$ and $b - A\mathcal{D}$ overlap in at least 90 percent of their hypervolume. So we have: $Pr_{v \in_U b - A\mathcal{D}}[||P(v) - A^{-1}v||_\infty \leq 2\epsilon] > 1/2$, from which $Pr_{x \in_U \mathcal{D}}[||P(b - Ax) - A^{-1}(b - Ax)||_\infty \leq 2\epsilon] > 1/2$ and $Pr_{x \in_U \mathcal{D}}[||x + P(b - Ax) - A^{-1}b||_\infty \leq 2\epsilon] > 1/2$. Thus, we have $Pr_{x \in_U \mathcal{D}}[||P(b) - (x + P(b - Ax))||_\infty \leq 2\epsilon] \leq 1/2$. So the probability that P can pass the self-check part of the test is at most $\frac{1}{2}^{\log_2 \frac{2}{\beta}} = \frac{\beta}{2}$ and the overall probability P can pass the test is at most β. □

4.3 Matrix Inversion

In this section we design a checker for programs P which output an approximation to the inverse of a nonsingular $n \times n$ matrix A. We use the approximate version of

Freivald's test to check that $||A(P(A)) - I||_\infty$ is small and from this, we make inferences about $||P(A) - A^{-1}||_\infty$.

```
Program Matrix Inversion Approximate Check($P, A, \epsilon, \beta$)
Compute $\hat{A} = \frac{A}{||A||_\infty}$
Call Matrix Multiplication Approximate Check($\hat{A}, P(\hat{A}), I, \epsilon, \beta$)
Output the output of the approximate matrix multiplication check
```

Theorem 3 *The above protocol is an* $(\frac{\epsilon}{n}, \frac{\sqrt{n}\epsilon}{\gamma})$*-approximate checker for matrix inversion, where* γ *is as in section 4.2, under the* $\|\cdot\|_\infty$ *norm.*

Proof. The proof is in the appendix. □

4.4 Computation of the Determinant

Kannan [10] shows a checker for the determinant when the program claims to compute it exactly. We show under what conditions we can check programs that claim to approximate the determinant, using a modification of his method. Let A be an $n \times n$ nonsingular matrix, and let γ be as above. Let $A_{i,j}$ be the matrix obtained by removing the i^{th} row and the j^{th} column from the matrix A, and let a_i be the i-th row of A. Let P be a program claiming to approximate the determinant. Let $v_P =< P(A_{1,1}), -P(A_{2,1}), \ldots, (-1)^{n+1}P(A_{n,1}) >^T$. The following is a checker for P, with matrix A.

```
Program Determinant Approximate Check($P, A, P(A), \epsilon, \beta$)
Consistency test
Repeat $O(\log(\frac{1}{\beta}))$ times:
    For $i = 1, \ldots, n$, choose random $\lambda_i \in \{-1, 1\}$
    Let $w^T = \Sigma_i \ \lambda_i a_i$
    Test that $|w^T v_P - \lambda_1 P(A)| \leq \epsilon |P(A)|$
Recursive test
    Let $j$ be such that $|P(A_{j,1})|$ is maximal
    Call Determinant Approximate Check($P, A_{j,1}, P(A_{j,1}), \epsilon, \beta$)
Output PASS $P(A)$ iff all tests pass.
```

Theorem 4 *The above protocol is an* $(\epsilon' n^{\frac{-3}{2}}, 2\epsilon' n^2 \gamma^{-1})$*-approximate checker for the determinant, where* $\epsilon' = \epsilon |det(A)|$.

Proof. The proof is in the appendix. □

4.5 Another Approximate Determinant Checker

In this section we consider ways to extend the self-tester and corrector for determinant presented in [4] to programs which work over the reals and claim only to approximate the determinant. The tester/corrector in [4] works as follows: on a program P and an input matrix A, it picks a random nonsingular matrix M with known determinant (using method of [13]) and it tests that $P(M) = det(M)$; then it corrects, by computing $\frac{P(MA)}{det(M)}$ for a random matrix M.

Note that this tester/corrector requires the use of a matrix multiplication routine which does not guarantee the little-oh property. However, they solve this by using Freivald's test and assuming that both the matrix multiplication and the determinant are part of a library that is being checked. To adjust the method in [4] to checking approximate computations of the determinant over the reals, we need to get around two major problems. The first is that the method shown in [13] does not easily adapt to uniformly generating matrices with known determinant over any simple subset of the real matrices. The second problem is that if we test program P over matrices belonging to distribution $\mathcal{D}$, then the self-correction done over $A\mathcal{D}$ may not be valid. Our solution is to precondition A by multiplying it by a matrix Q of known determinant. (Q can be for example a lower triangular matrix, as in the case of Gauss-Seidel preconditioning (see [9]).) We then choose a distribution $\mathcal{D}$ such that $\sum_{M \in \mathbb{R}^{n \times n}} |Pr_{M_1 \in \mathcal{D}}[M = M_1] - Pr_{M_2 \in QA\mathcal{D}}[M = M_2]|$ is small.

The approximate self-tester and corrector is the following:

```
Program Determinant Approximate Self-Test and Correct($P, A, \epsilon, \beta$)
Approximate Self-Test
    Repeat $O(\log \frac{1}{\beta})$ times
        Choose $M \in \mathcal{D}$ with known determinant
        and test that $\frac{|P(M) - det(M)|}{|det(M)|} < \epsilon$
    Output FAIL $P$ if any of the tests fails.
Approximate Self-Correct
    Repeat $O(\log \frac{1}{\beta})$ times
        Pick $M \in \mathcal{D}$ with known
        determinant and compute $\frac{P(QAM)}{det(Q)det(M)}$
    Output the median value.
```

Theorem 5 *The above approximate self-tester/corrector will successfully test and correct programs which output the determinant over D with relative error less than ϵ, with high probability.*

4.6 Approximate Self-Testers for Generalized Group Homomorphisms

Gemmell et al. [8] show approximate self-testers for functions that generalize linear functions on finite groups. We prefer to think of the property of the function as *generalized homomorphism*, rather than generalized linearity, because the group operators in the domain and range are not necessarily addition. Gemmell et al. [8] show that their tester works when the domain of the function is a finite abelian group, G_1 and the range, G_2, is isomorphic to either Z_n or Z. Although they do not state it explicitly, their approximate tester also applies to the case where the range is any linear metric space.[†] We extend their tester to the case when the domain is an infinite cyclic group $(G_1, \circ_{G_1})$ and we test over some finite subdomain. As with the tester in [8], the range $(G_2, \circ_{G_2})$ may be either a linear metric space or a space isomorphic to Z_n for some n. We use the following definitions:

Definition 8 *For any a and b in a metric space G_2, we say $a =_\Delta b \iff ||a \circ_{G_2} b^{-1}|| \leq \Delta$.*

Definition 9 *A function f mapping a group $(G_1, \circ_{G_1})$ to a group $(G_2, \circ_{G_2})$ is Δ_1-*approximately homomorphic *if $\forall a, b \in G_1, f(a \circ_{G_1} b) =_{\Delta_1} f(a) \circ_{G_2} f(b) \circ_{G_2} E(a, b)$, where the time complexity of computing $E : G_1 \times G_1 \rightarrow G_2$ is asymptotically less than that for computing f.*

We call a function a homomorphism if it is 0-approximately homomorphic. Notice that our definition of a homomorphic function encompasses a much larger class of functions than the traditional definition. We make use of the following approximate self-tester for generalized linear functions on domain G_1. This self-tester is from [8]. For concreteness, they assume the domain G_1 is equal to Z_n, although they point out that, with simple modifications, the tester works for generalized homomorphisms over finite abelian groups.

[†]By linear metric space, we mean $\forall x, ||x \circ_{G_2} x|| = 2||x||$.

```
Program Approximate General Homomorphism Self-Test($P, \Delta, \epsilon, \beta$)
(over finite group $G_1$)
Approximate Linearity Test
    Repeat $O(\frac{1}{\epsilon}\log\frac{1}{\beta})$ times
        Pick random $x, y \in G_1$ and test that $P(x \circ_{G_1} y) =_\Delta P(x) \circ_{G_2} P(y) \circ_{G_2} E(x,y)$
    Reject $P$ if the test fails more than $\epsilon/2$ fraction of the time.
Approximate Neighborhood Tests
    to be performed only if the range is isomorphic to $Z_n$
    Let $a = \lfloor\sqrt{m}\rfloor$
    Repeat $O(\log\frac{1}{\beta})$ times
        Pick random $x \in D$ and test that
            $P(x \circ_{G_1} 1) =_\Delta P(x) \circ_{G_2} f(1)$
            $P(x \circ_{G_1} a) =_\Delta P(x) \circ_{G_2} f(a)$
    Reject $P$ if test fails more than 1/4th of the time.
```

Theorem 6 *[8] The above protocol is an $(\epsilon/12, (4k+1)\sqrt{\epsilon}, \frac{\Delta-2\Delta_1}{3}, (1+\frac{5}{2^k})\Delta)$-approximate self tester for f where f is a Δ_1−approximate generalized homomorphism on domain Z_m.*

We convert the above approximate self-tester to an approximate self-tester for generalized homomorphisms over infinite cyclic groups. We test over a finite, evenly spaced subset of the domain. We describe a tester for a Δ_1-approximately *linear* function f mapping $(Z,+)$ to $(Z,+)$. Note that the tester we describe applies to the more general case of a function which maps any infinite cyclic group into any linear metric space; for notation's sake, we use Z as both the domain and the range. We will test on a restricted subdomain $\{0,\ldots,b-1\}$. We assume we know both $f(0)$ and $f(-b)$. Note that, once we prove that we can test on $\{0,\ldots,b-1\}$ then we can test on any arbitrary finite set of evenly spaced points by shifting and scaling.

Given a Δ_1-approximately linear function f over Z such that $\forall x,y \in Z, f(x+y) =_{\Delta_1} f(x)+f(y)+E(x,y)$, we define f', a $2\Delta_1$-approximately linear function over $(Z_b, +_b)$ such that $\forall x_b \in Z_b, f'(x_b) = f(x)$, where $\forall x \in \{0,\ldots,b-1\}$, x_b refers to x's representation modulo b and $+_b$ refers to addition modulo b. To see that f' is a $2\Delta_1$-approximately linear function, note that $\forall x_b, y_b \in Z_b, f'(x_b +_b y_b) =_{2\Delta_1} f'(x_b) + f'(y_b) + E'(x_b,y_b)$, where $E'(x_b,y_b) = E(x,y)$ if $x+y<b$, and $E'(x_b,y_b) = E(x,y) + E(x+y,-b) + f(-b)$ if $x+y \geq b$.

In order to test f over the interval $\{0,\ldots,b-1\}$, we simply test f' over the domain Z_b.

```
Program Approximate General Homomorphism Self-Test($P, \Delta, \epsilon, \beta$)
(Over $Z$)
Call Approximate General Homomorphism (over $Z_b$) Self-Test on $f'$
```

Theorem 7 *The above protocol is an* $(\epsilon/12, (4k+1)\sqrt{\epsilon}, \frac{\Delta-4\Delta_1}{3}, (1+\frac{5}{2^k})\Delta)$*-approximate self tester for f where f is a Δ_1- approximate generalized homomorphism mapping the integers into a linear metric space.*

Observation. The abover tester also applies directly to homomorphisms on the rationals (under addition) as well as to homomorphisms mapping domains isomorphic to $(Q, +)$. The subdomain that we test over is any finite set of evenly spaced points.

Corollary 8 *The above tester is an* $(\epsilon/12, 21\sqrt{\epsilon}, 0, 5)$*-approximate self tester for the quotient function over the integers:* $f_R(x) = \lfloor \frac{x}{R} \rfloor$ *where R is any fixed divisor.*

4.7 An Approximate Self-tester/corrector for Trigonometric Functions

Here we combine the ideas of [5] with the above extension of the approximate self-tester of [8] to obtain an approximate self-tester and corrector for the functions sin and cos over rational multiples of π. Let $D = \{\frac{2\pi l}{4k} : 0 \leq l < 4k - 1\}$ for some integer k. Define the operator $+'$ as follows: $\forall x = \frac{2\pi l}{4k} \in D$, let $x +' \frac{3\pi}{2} = \frac{2\pi}{4k}((l + 3k)(mod 4k))$. The reason we want this latter definition is because we will be needing to compute $sin(x) = cos(x + \frac{3\pi}{2})$.

Let GL^2_R be the group of 2×2 rotation matrices. In [5] the authors define the map $f : Q \rightarrow GL^2_R$ to be

$$f(x) = \begin{pmatrix} cos(x) & sin(x) \\ -sin(x) & cos(x) \end{pmatrix}.$$

f is a homomorphism between Q and GL^2_R. If we define a metric for the range which is invariant under matrix rotation operations, then the previous section on approximate self-testers for generalized homomorphisms on infinite cyclic groups provides us with an approximate self-tester for f over subdomain D. One metric we can use is: $dist(A, B) = sup_{i,j}\{(AB^{-1} - I)_{i,j}\}$.

We can use the approximate tester for f to achieve approximate self-testers for both sin and cos by means of the trigonometric identity $sin(x) = cos(x + \frac{3\pi}{2})$.

The approximate self-tester we propose for cos works as follows: Given a program P for cos, construct $P'(x)$, a program for $f(x)$:

$$P'(x) = \begin{pmatrix} P(x) & P(x +' \frac{3\pi}{2}) \\ -P(x +' \frac{3\pi}{2}) & P(x) \end{pmatrix}$$

The previous section gives an $(\epsilon/12, (4q + 1)\epsilon, \Delta/3, (1 + \frac{5}{2^q})\Delta)$-approximate self-tester for program P' which claims to compute f:

deal with this problem by checking the programs on suitable sets of random instances, which should not give rise to huge roundoff errors, with high probability.

Finally, we want to point out that this paper lays a foundation for a library of theoretically and practically efficient checkers for approximate computations.

Acknowledgment. We would like to thank W. Kahan and D. Randall for very helpful conversations.

References

[1] R. Beigel. Personal communication to Manuel Blum; see [2].

[2] M. Blum. Designing Programs to Check their Work. ICSI TR-88-009, 1988.

[3] M. Blum, S. Kannan. Designing Programs that Check their Work. In *Proc. 21st ACM Symposium on Theory of Computing*, 1989.

[4] M. Blum, M. Luby and R. Rubinfeld. Self-Testing/Correcting with Applications to Numerical Problems. In *Proc. 22nd ACM Symposium on Theory of Computing*, 1990.

[5] R. Cleve, M. Luby. A Note on Self-Testing/Correcting Methods for Trigonometric Functions. ICSI TR-90-032, July 1990.

[6] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progression. In *Proc. 19th Annual ACM Symposium on Theory of Computing*, pages 1–6, Berkeley, CA, 1987.

[7] R. Freivalds. Fast probabilistic algorithms. *Springer Verlag Lecture Notes in CS, Vol.74*, 57-69, 1979.

[8] P. Gemmell, R. Lipton, R. Rubinfeld, M. Sudan and A. Wigderson. Self-Testing/Correcting for Polynomials and for Approximate Functions. In *Proc. 23rd ACM Symposium on Theory of Computing*, 1991.

[9] G. H. Golub and C. F. van Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore, Md, 1983.

[10] S. Kannan. Program Checkers for Algebraic Problems. ICSI TR-89-064, 1989.

[11] R. Kannan, A.K. Lenstra, L. Lovasz. Polynomial factorization snd nonrandomness of bits of algebraic and some transcendental numbers. In *Proc. 16th ACM Symposium on Theory of Computing*, 1984.

[12] R. Lipton. New Directions in Testing. In *Distrib. Comput. and Cryptography, DIMACS Series on Discrete Mathematics and Theoretical Computer Science, pages 191-202*, 1991.

[13] D. Randall. Efficient random Generation of Invertible Matrices. personal communication.

[14] R. Rubinfeld. A mathematical theory of Self-Checking, Self-Testing, and Self-Correcting Programs. PhD Thesis, UC Berkeley, and ICSI TR-90-054, 1990.

A Appendix

Proof of Theorem 3

Proof. First, suppose that $||P(\hat{A}) - \hat{A}^{-1}||_\infty < \frac{\epsilon}{n}$. Then $||\hat{A}P(\hat{A}) - I||_\infty = ||\hat{A}(P(\hat{A}) - \hat{A}^{-1})||_\infty < ||\hat{A}||_\infty \frac{\epsilon}{n} \leq \epsilon$. So P with $\hat{A}$ will pass with high probability.

Second, suppose that $||P(\hat{A}) - \hat{A}^{-1}||_\infty \geq \frac{\sqrt{n}\epsilon}{\gamma}$. Then $||\hat{A}P(\hat{A}) - I||_\infty = ||\hat{A}(P(\hat{A}) - \hat{A}^{-1})||_\infty \geq \gamma \frac{\sqrt{n}\epsilon}{\gamma} = \sqrt{n}\epsilon$. So P with $\hat{A}$ will fail with high probability. □

Proof of Theorem 4

We prove Theorem 4 via two lemmas.

Lemma 12 *If program P with $n \times n$ matrix A passes the checker, with a given ϵ, with probability $\geq 1 - \beta$, then*

$$|\frac{det(A) - P(A)}{det(A)}| = O(n^2\gamma^{-1}\epsilon)$$

Proof. We denote by $Adj(A)$ the adjoint matrix of A and by v_A the vector $< P(A), 0, \ldots, 0 >^T$. If P with A passes the consistency test with ϵ, with high probability, then

$$\forall i \ \ |a_i \cdot v_P - P(A) \cdot [i = 1]| \leq \epsilon |P(A)|,$$

therefore, $\|Av_P - v_A\|_\infty \leq \epsilon |P(A)|$, from which $\|A^{-1}Av_P - A^{-1}v_A\|_\infty \leq \epsilon |P(A)| ||A^{-1}||_\infty$, so that

$$||v_P - P(A) \cdot (A^{-1})^1||_\infty \leq ||A^{-1}||_\infty \cdot \epsilon \cdot |P(A)|$$

and

$$||v_P - Adj(A)^1 + (det(A) - P(A)) \cdot (A^{-1})^1||_\infty \leq ||A^{-1}||_\infty \cdot \epsilon \cdot |P(A)|.$$

Let j be such that $|P(A_{j,1})|$ is maximal. We have

$$|\frac{det(A_{j,1})}{det(A)}(det(A) - P(A))| \ \leq \ ||A^{-1}||_\infty \cdot \epsilon \cdot |P(A)| \ + \ |(Adj(A))_{j,1} \ - \ P(A_{j,1})|.$$

Let β_n be the maximum (absolute value of) relative error with which P with an $n \times n$ matrix can pass with probability at least $1 - \beta$. Then

$$\beta_n \ \leq \ \frac{||A^{-1}||_\infty \cdot \epsilon \cdot |P(A)|}{|det(A_{j,1})|} \ + \ \beta_{n-1}$$

Because the entries of A are chosen from $[-1, 1]$, we know that $|det(A_{j,1})| = \Omega(\frac{|det(A)|}{n})$. We assume an upper bound of $\frac{1}{\gamma}$ on $||A^{-1}||_\infty$ and so we get:

$$\beta_n \leq \beta_{n-1} + O(n\epsilon \frac{|P(A)|}{\gamma |det(A)|}).$$

By solving the recurrence, and observing that $\frac{|P(A)|}{|det(A)|} \leq 2$, we get that

$$\beta_n = O(\frac{n^2\epsilon}{\gamma}).$$

□

Lemma 13 *If program P $(\epsilon n^{-3/2})$-approximates the determinant, then P with A will pass the consistency test with high probability.*

Proof. If P ϵ-approximates the determinant, we know that $\|e_A\|_\infty \leq \epsilon$, where e_A is an n-vector whose i-th entry is $P(A_{ii}) - det(A_{ii})$. If we now let D^1 be the n-vector whose i-th entry is $det(A_{i,1})$, and w_A the n-vector $< Det(A), 0, \ldots, 0 >^T$, we have that

$$||A \cdot v_P - A \cdot D^1|| \leq ||A||_\infty \cdot \epsilon$$

Since $AD_1 = w_A$, and using the triangle inequality, we have that

$$\|Av_P - v_A\|_\infty \leq \|Ae_A\|_\infty + \|w_A - v_A\|_\infty \leq ||A||_\infty \cdot \epsilon + \epsilon \; < \; (||A||_\infty + 1) \cdot \epsilon$$

Therefore, for random $\lambda_i \in \{-1, 1\}, i = 1, \ldots, n$

$$|(\Sigma_i \lambda_i a_i) \cdot v_P - \lambda_1 \cdot P(A)| \leq \sqrt{n} \cdot \epsilon \cdot (||A||_\infty + 1)$$

with high probability.
Using $||A||_\infty = O(n)$, we get:

$$|(\Sigma_i \lambda_i a_i) \cdot v_P - \lambda_1 \cdot P(A)| \leq n^{3/2} \cdot \epsilon$$

□

Proof of Theorem 11

Proof. If $\forall y$, $P_2(y) =_{\epsilon_1 - \epsilon_{12}} f_2(y)$ then $\forall x$, $P_1(x) =_{\epsilon_1} f_1(x)$, and, in turn, P_1 with $r_{21}(y)$ passes the test with high probability.
If $P_2(y)$ is not within $\epsilon_2 + \epsilon_{21}$ of $f_2(y)$, then $P_1(z)$ is not within ϵ_2 of $f_1(z)$ (since $f_2(y)$ is guaranteed to be within ϵ_{21} of $f_1(r_{21}(y))$. This, in turn, means that P_1 with $r_{21}(y)$ must fail, with high probability. □