

Floating-point L^2 -approximations to functions

Nicolas Brisebarre

Université de Saint-Étienne - LaMUSE,
23, Rue Dr Paul Michelon, 42023 Saint-Etienne Cedex 2,
nicolas.brisebarre@ens-lyon.fr

Guillaume Hanrot

INRIA - LORIA, Projet CACAO
Bâtiment A, 615, rue du jardin botanique, F-54602 Villers-lès-Nancy cedex,
hanrot@loria.fr

Abstract

In the present paper, we investigate the approximation of a function by a polynomial with floating-point coefficients; we are looking for the best approximation in the L^2 sense. Finding a best polynomial L^2 -approximation with real coefficients is an easy exercise about orthogonal projections. However, truncating the coefficients to floating-point numbers, which is needed for further computations, makes the approximation way worse. Hence, we study the problem of computing best approximations under the constraint that coefficients are floating-point numbers. We show that the corresponding problem is NP-hard, by reduction to the CVP problem.

We investigate the practical behaviour of exact and approximate algorithms for this problem. The conclusion is that it is possible in a short amount of time to obtain a relative or absolute best L^2 -approximation. The main applications are for large dimension, as a preliminary step of finding L^∞ -approximations and for functions with large variations, for which relative best approximation is by far more interesting than absolute.

1. Introduction

Computing elementary or special functions is a central topic in modern computer arithmetic. One common way of doing this is, given a continuous real-valued function f , to split the domain of f in sufficiently many small intervals; and then, over each of those intervals I , to compute a polynomial

P_I such that the distance between P_I and f over I is less than the accuracy expected. That distance is usually measured with the supremum norm (or L^∞ norm or absolute error)

$$\|P_I - f\|_\infty = \sup_{x \in I} |P_I(x) - f(x)|,$$

or the relative error

$$\|P_I - f\|_{\text{rel}} = \sup_{x \in I} \frac{1}{|f(x)|} |P_I(x) - f(x)|.$$

The classical theorem of Stone-Weierstraß (see eg. [5]) tells us that, for any $\varepsilon > 0$, we can find a polynomial such that $\max_{x \in I} \|f(x) - P(x)\|_\infty \leq \varepsilon$, but it gives no way of computing it. Remes' algorithm [18] gives an efficient way of computing this polynomial; however, this best approximant will not have machine number coefficients, except in very special cases. Of course, one can choose to round the coefficients of the polynomial computed by Remes' algorithm so that they fit the imposed finite-precision arithmetic format but this is not at all optimal: we are interested in getting the best accuracy possible with an approximation having the least degree and the smallest number of bits for representing the coefficients possible and this naive rounding process may give very poor results in that direction. A recent work [4] has made progresses towards the computation of best approximations *with machine number as coefficients*.

In this paper, we propose to investigate approximation by polynomials with machine-number as coefficients, but in the L^2 sense, ie. we are trying to minimize an expression

$$\|f - P_I\|_2 = \left(\int_I (f - P_I)^2 d\mu \right)^{1/2}$$

for some positive measure $d\mu$ over I . Not only interesting in itself, a motivation for studying it is that very good L^2 -approximations may constitute good approximations with respect to L^∞ norm or relative error. This is very useful since in several applications, one is not interested in getting the best approximation but rather a good enough one and also since a better estimate beforehand of the optimal error may lead to a significant speedup of the method presented in [4].

Roadmap of the paper. Section 2 discusses how to find best L^2 weighted approximations with real coefficients; this includes best relative approximations. In Section 3, we introduce the problem we address in that article. In Section 4, we shall prove that finding the best polynomial (or, more generally, approximation over any family of L^2 -functions) with machine numbers as coefficients is a (general) instance of the so-called CVP problem. We shall review briefly algorithms to solve CVP and give practical results in Section 5. Finally, we shall give some examples and compare the L^2/L^∞ approaches in Section 6, before concluding our paper.

2. L^2 -approximation

Let I be an interval of \mathbb{R} , and $d\mu$ a positive measure on I ; the reader not aware of measure theory can safely assume that $d\mu = w(x)dx$ for some positive weight function $w(x)$. We define the classical scalar product on $H = L^2(I, d\mu)$ as

$$(f|g) = \int_I f(t)g(t)d\mu.$$

Using a weight function $w(x)$ allows one to “fine-tune” the accuracy of the approximation in some parts of the interval. For instance if $I = [-a, a]$, the classical kernel $w(x) = (a^2 - x^2)^{1/2}$ will make the approximation sharper in the center of the interval, and more sloppy close to the bounds.

Let now $(e_i)_{0 \leq i \leq n}$ be a linearly independent family of elements of H , and put $E = \oplus_{i=0}^n \mathbb{R}e_i$. The following Lemma is elementary.

Lemma 1 *Let $f \in H$. The element $p_E(f)$ of E which is closest to f in the L^2 sense has coordinates over the basis (e_i) given by $G^{-1}V$, where G is the Gram matrix $((e_i|e_j))_{0 \leq i, j \leq n}$ and V the vector $((f|e_i))_{0 \leq i \leq n}$.*

In this classical setting, the problem under consideration is thus easy; in the sense that it is re-

duced to $O(n^2)$ computations of integrals and one linear system solution.

In the sequel, we shall denote by $d_E(f) := d(f, E)$ the L^2 distance from f to the space E i.e. $(\int_I (f - p_E(f))^2 d\mu)^{1/2}$. Note that this is a lower bound for the L^2 distance of f to any element of E .

Remark 1 *The “abstract” point of view developed above has the interest of allowing applications which are not restricted to finding best absolute polynomial approximations. This classical setting is obtained by taking $e_i = x^i$, and $d\mu = dx$ the Lebesgue measure, and the reader should probably keep this in mind as a roadmap.*

However, dealing with the problem in general-ity allows one to find approximation by trigonometric polynomials by taking $e_i = \cos(ix)$, or to find polynomial approximation using only certain monomials by taking $e_i = x^{k_i}$ where $(k_i)_{0 \leq i \leq n}$ is a finite strictly increasing sequence of natural integers, or to find best relative polynomial approximations to a function f , over an interval where f has no zero over I by taking $d\mu = dx/f(x)^2$, since then

$$\int_I (P(x) - f(x))^2 d\mu = \int_I \left(\frac{P(x)}{f(x)} - 1 \right)^2 dx.$$

These generalizations are obtained with the same algorithm; one does just need to compute slightly different integrals.

3. Floating-point approximations

In the context of floating-point computations, however, approximations such as $p_E(f)$ have a major drawback: their coefficients are (exact) real numbers. Using such an approximation in the context of floating-point evaluation is a disaster, since regardless of the intrinsic quality of the approximation, a truncation is performed on the coefficient, which often introduces an error larger than the approximation itself.

In order to avoid such a truncation, a better idea is to incorporate this constraint in the search for an optimal approximation, as in [4]. In this paper we are thus interested in finding the best approximation with “floating-point numbers” as coefficients.

We model our problem FP-appx in the following way:

FP-appx $((e_i)_{0 \leq i \leq n}, f, d\mu, I)$. Given an interval I , a positive measure $d\mu$ over

I , functions $(e_i)_{0 \leq i \leq n} \in L^2(I, d\mu)$, a function $f \in L^2(I, d\mu)$, find a vector $P = (p_i)_{0 \leq i \leq n} \in \mathbb{Z}^{n+1}$ which minimizes

$$\left\| \sum_{i=0}^n p_i e_i(x) - f(x) \right\|_2.$$

Note that it is easily proved that such a P exists. To see why this amounts to approximate by floating-point numbers, refer to the case where $e_i(x) = x^i 2^{-\varepsilon_i}$ for some integers ε_i ; in that case, we get an approximation by polynomials with coefficients of the form $m_i \cdot 2^{-\varepsilon_i}$ with $m_i \in \mathbb{Z}$. The choice of ε_i in practical applications allows one to control the size of the m_i , see Subsection 5.3.1.

4. Reduction to CVP

The problem described above can be restated in terms of lattices. Let L be the set $\{\sum_{i=0}^n u_i \cdot e_i, u_i \in \mathbb{Z}^{n+1}\}$. Then L , equipped with the restriction of the scalar product $(\cdot | \cdot)$ is a lattice of E of maximal rank, and the problem under consideration amounts to find the vector of L which is the closest to the vector with coordinates the coefficients of $p_E(f)$ in the basis e_i . In the sequel, we shall identify without notice a polynomial and the vector of its coefficients in a given basis, usually made clear by the context.

This problem is called CVP in the literature dealing with lattices, see e.g. [1] for a survey on the topic. In the present case, the problem is formulated in Gram form. Indeed, the classical form is

CVP(M, v). Given an $n \times n$ matrix M and a vector $v \in \mathbb{R}^n$, find the vector $x \in \mathbb{Z}^n$ such that $|Mx - v|_2$ is minimal, where $|(y_i)_{1 \leq i \leq n}|_2 = (\sum_{i=1}^n y_i^2)^{1/2}$.

whereas, in the present situation, we are trying to minimize the distance between a point of the lattice and the orthogonal projection of f on $\oplus_{i=0}^n \mathbb{R}e_i$. Lemma 1 shows that the latter is given by $G^{-1}V$. Since the L^2 -norm of any vector W of $\oplus_{i=0}^n \mathbb{R}e_i$ is $W^t G W$, we deduce our CVP-Gram form:

CVP-Gram(G, V). Given an $n \times n$ symmetric definite positive matrix G and a vector $V \in \mathbb{R}^n$, find the vector $X \in \mathbb{Z}^n$ which makes $(X - G^{-1}V)^t G (X - G^{-1}V)$ minimal.

The first form easily reduces to the second one, by taking $G = M^t M$ and $V = Gv$. Conversely, the second form can be reduced, at least numerically, to the first, by computing a Cholesky decomposition (a ‘‘square root’’) of the matrix G .

The CVP problem is a difficult problem, NP-hard even when allowing quite large approximation factors. We proceed to prove that this NP-hardness implies the NP-hardness of our problem $\mathbb{F}\mathbb{P}$ -appr. In fact, we now prove that our problem is indeed a general CVP-Gram, which means that any CVP instance can be reduced to it. We start by giving a precise formulation of the problem we are studying. The formulation given below is actually a bit less general than above (we restrict the e_i and f to be polynomials and assume that the measure $d\mu$ is given by its first $2n+1$ moments), in order to deal with objects that have a finite representation.

Proposition 1 *Let (u_0, \dots, u_n) be $n+1$ linearly independent vectors in \mathbb{R}^{n+1} , L the lattice they generate, and $v \in \mathbb{R}^{n+1}$ a vector. There exist a sequence of polynomials $(P_i)_{0 \leq i \leq n+1}$ with rational integer coefficients, a sequence $(\mu_i)_{0 \leq i \leq 2n}$ of rational numbers such that CVP-Gram(G, V) reduces to $\mathbb{F}\mathbb{P}$ -appr($(P_i)_{0 \leq i \leq n}, P_{n+1}, d\mu, \mathbb{R}$) where $d\mu$ is a positive measure over \mathbb{R} with $(\mu_i)_{0 \leq i \leq 2n}$ as its first $2n+1$ moments. Further, all those polynomials can be computed in polynomial time.*

Proof. Up to a (rational) change of basis, we can assume that the matrix $(g_{ij})_{0 \leq i, j \leq n}$ is diagonal. This is achieved in practice by Gram-Schmidt orthogonalization, which can be performed (over \mathbb{Q}) in polynomial time. We are thus looking for orthogonal polynomials Q_i for a measure, with given norm $\sqrt{g_{ii}}$. We start with the polynomials and build the moments of the corresponding measure.

First, put $g_{kk} = 1$ for $n+1 \leq k \leq 2n$. Put $\lambda_0 = g_{00}$ and $\lambda_i = g_{ii}/g_{i-1, i-1}$ for $i = 1, \dots, 2n$, and define Q_i by the recurrence relation

$$Q_0 = 1, Q_1 = x, Q_{i+1} = xQ_i - \lambda_i Q_{i-1}$$

for $i = 1, \dots, 2n$. Put $\mu_0 = \lambda_0$. Then, from

$$\int_{\mathbb{R}} Q_0 Q_1 d\mu = 0 = \int_{\mathbb{R}} x d\mu,$$

we deduce that $\mu_1 = 0$.

More generally,

$$0 = \int_{\mathbb{R}} Q_i Q_0 d\mu = \int_{\mathbb{R}} x Q_{i-1} d\mu - \lambda_i \int_{\mathbb{R}} Q_{i-2} d\mu,$$

which, by expanding the polynomials Q_i , allows to find μ_i in terms of the previous μ_j , for all $i = 1, \dots, 2n$. Then, for $k \leq l-1$ one can rewrite $x^k Q_l(x)$ as a linear combination of Q_{l-k}, \dots, Q_{l+k} , hence $\int_{\mathbb{R}} x^k Q_l(x) d\mu = 0$, which shows that Q_l is indeed orthogonal to Q_0, \dots, Q_{l-1} .

Since the g_{ii} are positive, the bilinear form induced by μ on the polynomials of degree $\leq 2n$ is positive definite, so that Hamburger's theorem [9, 10, 11] applies and there exists a positive measure μ over \mathbb{R} with first moments μ_i .

Finally, for $1 \leq i \leq n$, since Q_i is orthogonal to the Q_j for $j < i$, it is orthogonal to any polynomial of degree $< i$. Hence

$$\begin{aligned} \int_{\mathbb{R}} Q_i^2(x) d\mu &= \int_{\mathbb{R}} x^i Q_i(x) d\mu \\ &= \int_{\mathbb{R}} x^{i-1} Q_{i+1}(x) d\mu \\ &\quad + \lambda_i \int_{\mathbb{R}} x^{i-1} Q_{i-1}(x) d\mu \\ &= \lambda_i \int_{\mathbb{R}} Q_{i-1}^2(x) d\mu, \end{aligned}$$

which shows, by induction, that $\int_{\mathbb{R}} Q_i^2(x) d\mu = g_{ii}$. \square

This shows, in view of [21], that FP-appr is NP-hard; further, as a corollary from [14], we obtain

Corollary 1 *The problem FP-appr is NP-hard, even if we allow an approximation factor up to $n^{O(1/\log \log n)}$.*

In practice, fortunately, the situation is by far better. First, the problem is polynomial when the dimension is fixed. Second, the exponential algorithms behave quite well in our situation, where we are looking for approximations of small enough dimension (in any case, less than, say, 50).

5. A short review of algorithms for CVP

The previous section shows us that we need to study the behaviour of CVP algorithms in the context of our problem. Though there is already abundant theoretical and practical literature about the CVP [1], we include this section for the sake of completeness, but only give heuristic descriptions of the algorithms.

In a nutshell, algorithms for CVP can be decomposed into two distinct steps for which various solutions exist. The first step is a preprocessing

step, which computes a more or less strongly reduced basis of the lattice under study. The second one actually solves the CVP. Usually, the stronger the reduction, the more accurate or efficient is the second step.

5.1. Short review of CVP algorithms

5.1.1 A naive method

Let us start with an extremely naive method, which should probably not be used but serves as a good introduction. Lemma 1 tells us that the coordinates of $p_E(f)$ in the basis e_i are given by $G^{-1}V$. A simple idea is thus to compute $G^{-1}V$ and to round to the nearest integer the values that we have obtained in order to get a vector with integer coordinates in the basis (e_i) .

Lemma 2 *This method is optimal if the basis (e_i) is orthogonal.*

Proof. Write $(f_i) = G^{-1}V$. Then, since the basis is orthogonal, for any $x \in \mathbb{Z}^n$, the L^2 distance of $\sum_{i=0}^n x_i e_i$ to $p_E(f) = \sum_{i=0}^n f_i e_i$ is given by

$$\left(\sum_{i=0}^n (x_i - f_i)^2 \|e_i\|_2^2 \right)^{1/2},$$

which is minimal for $x_i = \lfloor f_i \rfloor, \forall i = 0, \dots, n$. \square

5.1.2 Babai's nearest plane algorithm

Babai's method [2] is a refinement of the previous method. In a very sketchy way, it can be described as follows: rounding each coefficient $M^{-1}v$ in the naive method induces an error. Thus, a better way is to round each coefficient one after the other, and to reintroduce the error coming from one rounding before performing the next.

More formally, this gives Algorithm 1.

If the input basis is of sufficient quality, we can deduce estimates on the quality of the output result. We quote the following classical result without a proof (this result is usually stated for an LLL-reduced basis, but adapts to the general case). The family $(e_i^*)_{0 \leq i \leq n}$ denotes the basis given by the Gram-Schmidt orthogonalization of the basis $(e_i)_{0 \leq i \leq n}$.

Theorem 1 *If we apply Algorithm 1 above to the FP-appr problem, we obtain a polynomial $P = \sum_{i=0}^n x_i e_i$ such that*

$$\|P - p_E(f)\|_2 \leq \frac{\gamma^{n+1}}{2\sqrt{\gamma^2 - 1}} \|\tilde{P} - p_E(f)\|_2,$$

Algorithm:Babai

Data: An $(n + 1) \times (n + 1)$ Gram matrix G ;
an $(n + 1)$ -vector V

Result: An approximation of
CVP-Gram(G, V)

```

begin
  Compute a Gram-Schmidt decomposition
  of  $G = B^t DB$ , with  $B$  orthogonal and  $D$ 
  diagonal;
   $V \leftarrow BV$ ;
  for ( $j = n; j \geq 0, j--$ ) do
     $X[j] \leftarrow \lfloor W[j]/D[j, j] \rfloor$ ;
    for ( $i = 0; i \leq n; i++$ ) do
       $W[i] \leftarrow$ 
       $W[i] - X[j]B[j, i]D[i, i]$ ;
    end
  end
  return  $X$ ;
end

```

Algorithm 1: Babai’s nearest Plane algorithm

for any $\gamma \geq \max_{0 \leq i \leq n-1} \|e_i^*\|_2 / \|e_{i+1}^*\|_2$ and \tilde{P} is the actual solution to the FP-approx problem.

A consequence is the fact that the most orthogonal the basis, the best Babai’s algorithm. This suggests again to use a preprocessing step, as described below.

5.1.3 Exact method

Exact methods, which are all a variant of an algorithm initially due to Kannan [15] consist in starting with an approximate solution, i.e. a vector w in the lattice such that $R := (v - w | v - w)$ is small. Then, the goal is to find a vector x in the ellipsoid $(v - x | v - x) \leq R$. This is done by enumerating points in this ellipsoid, or in a small superset of it. Note that once a good approximation is found, it can be used to restrict again the set of points to enumerate, since we are interested in a smaller ellipsoid. We do not give further detail, and refer to [1] for pseudo-code for this method. Let us simply point that the complexity of this approach is superexponential in the dimension d of the lattice, namely $d^{d/2e+o(d)}$ up to polynomial factors in the input size [12].

5.2. Short review of preprocessing

The accuracy (approximate methods) / efficiency (exact method) of those methods highly depend on the geometry of the basis of our lattice L .

For instance, the first algorithm becomes optimal if the basis is orthogonal; and the most orthogonal the basis, the best Babai’s algorithm performs. Finally, if the basis is more orthogonal, the ellipsoid in the exact method is much easier to enumerate. This leads to the idea that one should try to use the most orthogonal basis possible. This suggests to use a lattice basis reduction algorithm as a preprocessing step. In this paper, we chose to use LLL-reduced bases (see [16], computable in polynomial time) or Korkine-Zolotareff reduced bases (see [13], reasonably efficient for small dimensions); one of the reasons for this is the fact that software for computing those bases is widely available, eg. in NTL [20].

It should be pointed, however, that our experiments showed that, in the case of polynomial approximation, the bases are sufficiently orthogonal, and in practice Babai’s algorithm always gave the optimal solution. One might, however, expect that the situation changes if the dimension increases too much (say, above 50), but this does not seem to be the case for applications.

5.3. Tuning of the method

For applying the method, we now need to describe how one should choose the various free parameters in our description. Recall that in the case of polynomial approximation, we are trying to approximate a function f by a linear combination over \mathbb{Z} of the functions $2^{-\varepsilon_i} x^i$, $i = 0, \dots, n$,

$$f(x) \approx \sum_{i=0}^n m_i 2^{-\varepsilon_i} x^i.$$

We need to choose n , the size of the m_i and the ε_i . We postpone the choice of the ε_i , since they will serve to enforce the size chosen for the m_i .

5.3.1 Choosing n and the ε_i

In order to improve the quality of the approximation, one has the choice between increasing n and increasing the ε_i . This compromise should be carefully selected. Indeed,

- Increasing n has the effect to decrease the intrinsic L^2 -approximation error, ie. $\|f - p_E(f)\|_2$;
- Increasing the size of the ε_i decreases the “floating-point approximation” error, ie. $\|p_E(f) - \phi_E(f)\|_2$, where $\phi_E(f)$ is the solution to FP-approx.

As a consequence, playing on a single side is pointless, since the other error shall remain of the same order of magnitude, and the sum of the two errors in the global approximation scheme will not decrease.

A first remark is the fact that in practice, one is willing to use for $m_i 2^{-\varepsilon_i}$ some kind of machine floating-point number, either double precision, single precision, double extended, quad, or any combination of those (by taking $m_i 2^{-\varepsilon_i}$ to be the sum of two floating-point numbers, we can assume that m_i has size 106 bits, for instance). Since our algorithm only allows us to choose the ε_i , we need to relate the size of m_i and ε_i .

Heuristic 1 For all i , $m_i 2^{-\varepsilon_i}$ is close to the coefficient of x^i in $p_E(f)$.

This heuristic is true in practice as long as the corresponding coefficients do not get too small. In large degrees, with small coefficients, it might be wrong. In that case, one should replace in the heuristic $p_E(f)$ by the output of a first round of the algorithm: this modified heuristic allowed us to correctly guess the value of the ε_i to get a given number of bits for m_i in all cases we tried. A similar heuristic is used in the paper [3] for L^∞ approximation.

Heuristic 2 ε_i should be chosen of the form $C + \log_2 \|x^i\|_2$, for C not depending on i , and thus the size of m_i should be chosen of the form $\log_2 |(p_E(f)|x^i)| + C$.

Indeed, if the basis x^i were orthogonal, and if $p_E(f) = \sum_{i=0}^n \varphi_i x^i$, the error can be written as

$$\|P - p_E(f)\|_2^2 = \sum_{i=0}^n \left| \frac{m_i}{2^{\varepsilon_i}} - \varphi_i \right|^2 (x^i | x^i).$$

The i -th term in this sum is expected to be of the order of $2^{-2\varepsilon_i - 2} \|x^i\|_2^2$. Thus, ε_i should be taken as $\log_2 \|x^i\|_2 + C$ in order to minimize the global error. Now, the previous heuristic, again if the basis is orthogonal, gives

$$\log_2 m_i - \varepsilon_i \approx \log_2 |(p_E(f)|x^i)| - \log_2 (x^i | x^i).$$

In practice, we expect that the basis is “sufficiently” orthogonal for the heuristic above to give results close to optimal.

We can thus concentrate on finding which degrees n and total size of m_i yield the best approximation for a given total number of bits, ie. choosing m_i such that $\log_2 \|\varphi_E(f) - p_E(f)\| \approx \log_2 \|f - p_E(f)\|_2$.

Estimating $\log_2 \|\varphi_E(f) - p_E(f)\|$ is estimating the distance from a random point of \mathbb{R}^{n+1} to some lattice L . In the case of a random point and a random lattice, we can expect it to be of the order of the length of a shortest vector of the lattice, which by Minkowski’s first theorem is of the order of $(\text{disc } L)^{1/(n+1)}$.

Heuristic 3 Let G' be the Gram matrix $(x^i | x^j)$. We expect that

$$-\log_2 \|\varphi_E(f) - p_E(f)\|_2 \approx \frac{\sum_{i=0}^n \varepsilon_i}{n+1} - \frac{\log_2 \det(G')}{2(n+1)}.$$

This gives the choice of the average value of ε_i , from which the two previous heuristics allow to deduce heuristics for $\log_2 m_i$, in order to achieve a given error.

5.4. Numerical issues

The main numerical issues of our algorithm are

- the computation of orthogonal bases which is required at almost every step of the method,
- the problem of computing accurately enough the integrals that define the matrix G ,
- the definition of the problem itself, namely the fact that we shall have to replace exact real numbers (the value of the integral) by floating-point (or, in practice, integer) approximations.

We shall take the second item for granted, eg. by using [7]. The first item is a classical topic and has been extensively studied, see for instance [17]. We shall thus focus on the last point. The meaning of this is that we replace the exact values of G and V by approximations \tilde{G} and \tilde{V} . A first (trivial) remark is that the problem is already ill-defined in dimension 1 : finding the element of $\mathbb{Z}\pi$ closest to $\pi/2$ is highly subject to numerical approximation. However, the *distance* to $\mathbb{Z}\pi$ is well-defined: the solution found might change, but we are switching from an optimal solution to another solution which is arbitrarily close of being optimal. In the example above, depending on the roundings, we might find either 0 or π , both being admissible answers to the original problem.

Theorem 2 For M a matrix, we define $\|M\|$ to be the maximum of absolute values of its coefficients,

and similarly for a vector. Assume that $\max(\|G - \tilde{G}\|, \|V - \tilde{V}\|) \leq \varepsilon$, and that $(n+1)\varepsilon\|\tilde{G}^{-1}\| < 1$. Then, for all $x \in \mathbb{Z}^{n+1}$, we have

$$\left| (x - G^{-1}V)^t G(x - G^{-1}V) - (x - \tilde{G}^{-1}\tilde{V})^t \tilde{G}(x - \tilde{G}^{-1}\tilde{V}) \right| \leq \varepsilon C,$$

where C is an explicit constant depending only on x , n , \tilde{G} , \tilde{V} .

Proof. Left to the reader. \square

This theorem shows that a “good” vector for the approximate problem is not too far away from a “good” vector for the exact problem, and (by symmetry) vice-versa. The drawback is that the quality of the approximation depends on x . However, since x lies in a bounded region of \mathbb{R}^n , one can in theory remove this dependency, or check once x is computed that the precision is sufficient. Using the theorem (and the theorem where we exchanged the roles of G, V and \tilde{G}, \tilde{V}) shows that by increasing the precision we can come as close as we want to the actual optimal.

6. Some experiments, comparison with L^∞ -approximation

We have implemented the various combinations of methods described above and tested them for the problem described above on several functions. The algorithms have been implemented in C, using NTL-5.4[20], GMP-4.2.1 [8] and MPFR-2.2.0[19]. The scalar products have been computed using GP/pari function `intnum`, but we plan to replace it with software yielding guaranteed results, such as the CRQ library [7].

These implementations of the LLL algorithm may present numerical instability [17], but this is not expected on matrices of the kind we reduce, in the dimension useful for applications. In any case, this does not compromise the optimality of the results computed; simply, the preprocessing is of lesser quality than expected, and thus the computation time slightly larger.

All these computations, even in larger dimension, have taken less than a second on an Intel Pentium M 2GHz.

Example 1 We consider the function $x \mapsto \sin(\pi\sqrt{x})/(\pi\sqrt{x})$ on the interval $[0, 1]$ and we search for a degree-8 polynomial approximation with single precision floating-point coefficients.

We get an L^2 error equal to $1.883 \dots \cdot 10^{-21}$, that we compare to $2.562 \dots \cdot 10^{-17}$ which is the error given by the rounded projection i.e. the polynomial obtained after having rounded to the nearest the coefficients of the real optimal L^2 -approximation: we get a factor 10000 improvement.

We obtain an absolute error equal to $1.345 \dots \cdot 10^{-10}$, that we have to compare to $1.002 \dots \cdot 10^{-8}$ which is the error provided by the rounded minimax (i.e. the polynomial obtained after having rounded to the nearest the coefficients of the real optimal L^∞ -approximation): we obtain an improvement by a factor 100.

Example 2 We consider the arctan function on $[-1, 1]$. Then, we search for a polynomial of the form $p(x) = x + x^3(p_0 + p_1y + \dots + p_{22}y^{22})$ where $y = x^2$ and the p_i are double precision floating-point numbers. This is a practical example from [6, Chap. 10]

We immediately obtain an approximant giving rise to a relative error equal to $7.28 \dots \cdot 10^{-19}$ which improves by a factor 32 the relative error given by the rounded minimax which is equal to $2.28 \dots \cdot 10^{-17}$.

There is currently another lattice reduction based approach [3] developed for tackling the problem of getting very good L^∞ -approximation. On the few experiments we did, we noticed that this approach sometimes leads to better results than ours but the factor between the errors they reach and ours is not large, even in degrees around 25. That approach has the drawback to require as input the computation of a minimax approximation, given by Remes’ algorithm, whereas our approach is entirely done at almost no cost. Moreover, there are issues that the approach of [3] is not yet able to address but which can be treated with the method developed here. For instance, we already noticed that we can tackle with relative error approximation. We are also able to compute approximations in which the values of some coefficients are fixed. Finally, we can directly adapt our method to compute approximations to two-variable functions.

7. Conclusion and further work

We have presented a set of methods for computing best (absolute or relative) L^2 floating-point approximations, either by polynomials, or more generally by any finite linear combination of functions, such as trigonometric polynomials. These

methods can be seen as a further tool in the toolbox available to the “computer arithmetician” to compute approximations with floating-point (and more generally machine-number) coefficients.

We have reviewed the main features of this tool: efficiency (especially in large dimension), ability to deal with relative error, and versatility, utility as a preprocessing step to speed up L^∞ or relative error computations.

The fact that the best vector produced by the L^2 method seems in practice to be within a small factor of the L^∞ solution suggests an improved strategy. Denote by b the best L^2 -approximation that we have obtained. By using a straightforward adaptation of the optimal algorithm, we can list all vectors v for which

$$\|v - p_E(f)\|_2 \leq \beta \|b - p_E(f)\|_2, \quad (1)$$

for some constant $\beta > 1$.

Then, we can compute the corresponding L^∞ norm of all those vectors, and look for the best of them. The main trouble is the fact that the number of vectors will grow roughly as ¹

$$\left(\frac{\beta \|b - p_E(f)\|_2}{(\det(G))^{1/2n}} \right)^n;$$

If the ε_i are not chosen carefully enough, $\|b - p_E(f)\|_2$ may be much larger than $(\det G)^{1/2n}$. More intuitively, $\|b - p_E(f)\|_2$ is of the order of magnitude of the size of the largest vector in a “good” basis; if we have many “small” vectors in the basis, any “not too large” linear combination of those vectors can be added to b without changing significantly its L^2 -distance to $p_E(f)$. In that case, the set of vectors which verify (1) may be very large, and should be dealt with in a non-exhaustive way.

References

- [1] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger. Closest point search in lattices. *IEEE Transactions on Information Theory*, 48(8):2201–2214, 2002.
- [2] L. Babai. On Lovász’ lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986.
- [3] N. Brisebarre and S. Chevillard. Efficient polynomial L^∞ -approximations. 2007. This volume.
- [4] N. Brisebarre, J.-M. Muller, and A. Tisserand. Computing machine-efficient polynomial approximations. *ACM Transactions on Mathematical Software*, 32(2), June 2006.
- [5] E. W. Cheney. *Introduction to approximation theory*. AMS Chelsea Publishing, second edition, 1982.
- [6] M. Cornea, J. Harrison, and P. T. P. Tang. *Scientific Computing on Itanium-Based Systems*. Intel Press, 2002.
- [7] L. Fousse. CRQ, the correctly rounded quadrature library for crq, February 2006. <http://www.komite.net/laurent/soft/crq/>.
- [8] T. Granlund. GMP, the GNU multiple precision arithmetic library, version 4.2.1, May 2006. <http://www.swox.com/gmp/>.
- [9] H. Hamburger. Über eine Erweiterung des Stieltjesschen Momentenproblems, I. *Math. Ann.*, 81:235–319, 1920.
- [10] H. Hamburger. Über eine Erweiterung des Stieltjesschen Momentenproblems, II. *Math. Ann.*, 82:120–164, 1921.
- [11] H. Hamburger. Über eine Erweiterung des Stieltjesschen Momentenproblems, III. *Math. Ann.*, 82:168–187, 1921.
- [12] G. Hanrot and D. Stehlé. Tighter analysis of Kannan’s enumeration algorithm. 2007. Submitted.
- [13] C. Hermite. *Oeuvres complètes*. Gauthier-Villars, 1912.
- [14] S. S. I. Dinur, G. Kindler. Approximating-CVP to within almost-polynomial factors is np-hard. In *39th Annual Symposium on Foundations of Computer Science*, pages 99–111. IEEE Computer Society, 1998.
- [15] R. Kannan. Minkowski’s convex body theorem and integer programming. *Math. Oper. Res.*, 12(3):415–440, 1987.
- [16] A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Math. Annalen*, 261:515–534, 1982.
- [17] P. Nguyen and D. Stehlé. Floating-point LLL revisited. In *Proceedings of Eurocrypt 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 215–233. Springer-Verlag, 2005.
- [18] E. Remes. Sur un procédé convergent d’approximations successives pour déterminer les polynômes d’approximation. *C.R. Acad. Sci. Paris*, 198:2063–2065, 1934.
- [19] The Spaces project. MPFR, the multiple precision floating point reliable library, version 2.2.0. <http://www.mpfr.org>, 2005.
- [20] V. Shoup. NTL, a library for doing number theory, version 5.4. <http://shoup.net/ntl/>, 2005.
- [21] P. van Emde Boas. Another NP-complete problem and the complexity of computing short vectors in a lattice. Technical Report 81-04, Mathematische Instituut, University of Amsterdam, 1981.

¹this estimate follows from the approximation that the number of integral points within a region can heuristically be estimated by the volume of this region, and applying a linear transformation in order to map the lattice on \mathbb{Z}^n .