The Accurate Solution of Certain Continuous Problems Using Only Single Precision

CUCS-82-83

M. Jankowski

Institute of Informatics
University of Warsaw

H. Woźniakowski*

Institute of Informatics
University of Warsaw

and

Department of Computer Science
Columbia University

November 1983

*This research was supported in part by the National Science Foundation under Grant MCS-7823676.

Abstract.

A typical approach for finding the approximate solution of a continuous problem is through discretization with meshsize h such that the truncation error goes to zero with h. The discretization problem is solved in floating point arithmetic. Rounding-errors spoil the theoretical convergence and the error may even tend to infinity.

In this paper we present algorithms of moderate cost which use only single precision and which compute the approximate solution of the integration and elliptic equation problems with full accuracy. These algorithms are based on the modified Gill-Møller algorithm for summation of very many terms, iterative refinement of a linear system with a special algorithm for the computation of residuals in single precision and on a property of floating point subtraction of nearby numbers.

1. Introduction.

Suppose we wish to approximate the solution u of a continuous problem $u = S(f)$. Here S is an operator. For instance, $S(f)$ may denote the integral of a function f and $S(f)$ may denote the solution of an elliptic equation with a right-hand side f as illustrated below. A typical approach is to find a suitable discretization $u_h = S_h(f)$, where h is a discretization parameter, with a truncation error $O(h^p)$ for some positive p. The discretized problem $u_h = S_h(f)$ is solved in t digit floating point binary arithmetic. Due to rounding errors one computes $\bar{u}_h$ such that $u_h - \bar{u}_h = O(h^{-k}2^{-t})$ for some nonnegative k. Thus

$$(1.1) \qquad u - \bar{u}_h = O(h^p + h^{-k}2^{-t}).$$

For most algorithms that compute $\bar{u}_h$, the parameter k is positive. Thus if h tends to zero, the influence of rounding errors spoils the theoretical convergence and $u - \bar{u}_h$ may even tend to infinity with $h \to 0$ and fixed t. One may choose h in (1.1) so that to minimize the function $h^p + h^{-k}2^{-t}$. Then $h = h_0 = O(2^{-t/(p+k)})$ and

$$(1.2) \qquad u - \bar{u}_{h_0} = O(2^{-t\frac{p}{p+k}}).$$

For positive k, this means that one <u>cannot</u> guarantee the computation of a t digit approximation to the exact solution u although t digit arithmetic is used. This holds regardless of the smoothness of the problem $u = S(f)$. We illustrate this point by two simple examples.

Example 1.1: Integration.

Let $f: [0,1] \to \mathbb{R}$ be a once continuously differentiable function. Let

$$u = S(f) = \int_0^1 f(t)dt.$$

For $h = 1/n$ define

$$u_h = S_h(f) = h \sum_{i=1}^n f(ih)$$

which is the rectangle quadrature formula. Then $u - u_h = O(h)$ which corresponds to $p = 1$ in (1.1).

To compute u_h we apply the usual algorithm for summing of n numbers. Assuming for simplicity that $f(ih)$ is computed exactly we have

$$\bar{u}_h = h \sum_{i=1}^n f(ih)(1 + \varepsilon_i)$$

where $|\varepsilon_i| \le 1.06(n + 2 - i)2^{-t}$ whenever $(n+1)2^{-t} \le 0.1$, see [11]. Thus $u_h - \bar{u}_h = O(h^{-1}2^{-t})$ which corresponds to $k = 1$ in (1.1). For this case the optimal $h = h_0 = O(2^{-t/2})$ and

$$u - \bar{u}_{h_0} = O(2^{-t/2}).$$

It is possible to improve this estimate by applying the Gill-Møller algorithm for summation of n numbers, see [7]. Then as was proven in [5] we have

$$\bar{u}_h = h\,\Sigma_{i=1}^{n}\ f(ih)(1+\delta_i)$$

with $|\delta_i| \le (3 + n^2 2^{-t})2^{-t}$ whenever $(n+1)2^{-t} \le 0.1$. Thus

$$u - \bar{u}_h = O(h + 2^{-t} + h^{-2}2^{-2t}).$$

The optimal $h = h_0 = O(2^{-2/3t})$ and

$$u - \bar{u}_{h_0} = O(2^{-2/3t}).$$

This means that using the classical algorithm of summation one can compute a t/2 digit approximation to the integral of f whereas the Gill-Møller algorithm yields a $\frac{2}{3}t$ digit approximation provided these two algorithms use t digit arithmetic. ∎

Example 1.2: <u>Model Elliptic Equation.</u>

Let $f: (0,1) \to \mathbb{R}$ be a sufficiently smooth function. Let $u = S(f)$ be a solution of the one dimensional elliptic equation

$$u''(x) = f(x), \qquad x \in (0,1),$$

$$u(0) = g_0, \qquad u(1) = g_1,$$

for some constants g_0 and g_1. We discretize $u = S(f)$

by $u_h = S_h(f)$ where $h = \frac{1}{n+1}$ and u_h is a solution of the $n \times n$ linear system $L_h u_h = f_h$ with

$$L_h = \begin{bmatrix} 2, & -1 & & & \\ -1, & 2, & -1 & & \\ & \cdot & \cdot & \cdot & \\ & & \cdot & \cdot & \cdot \\ & & & \cdot & \cdot & \cdot \\ & & & -1, & 2, & -1 \\ & & & & -1, & 2 \end{bmatrix}, \qquad f_h = \begin{bmatrix} g_0 - h^2 f(h) \\ - h^2 f(2h) \\ \cdot \\ \\ -h^2 f(1-2h) \\ g_1 - h^2 f(1-h) \end{bmatrix}$$

Then $u(ih) - u_{h,i} = O(h^2)$, $i = 1,2,\ldots,n$, where $u_{h,i}$ is the ith component of u_n. This corresponds to $p = 2$ in (1.1).

To compute u_h apply, for instance, Gaussian elimination. Assuming for simplicity that $f(ih)$ is computed exactly, Gaussian elimination produces $\bar{u}_h$ which is the exact solution of a slightly perturbed matrix, i.e.,

$$(L_h + E_h)\bar{u}_h = f_h$$

with $\|E_h\|_\infty$ of order 2^{-t}. From this we have

$$u_h - \bar{u}_h = L_h^{-1} E_h \bar{u}_h.$$

Since $\|L_h^{-1}\|_\infty$ is of order h^{-2} then $\|L_h^{-1} E_h \bar{u}_h\|_\infty \le \|L_h^{-1}\|_\infty \|E_h\|_\infty \|\bar{u}_n\|_\infty = O(h^{-2} 2^{-t})$. We stress that $\|L_h^{-1} E_h \bar{u}_h\|_\infty$ is of order $h^{-2} 2^{-t}$ whenever E_h is <u>not</u> correlated to L_h^{-1} or $\bar{u}_h$. Thus

$\|u_h - \bar{u}_h\|_\infty = O(h^{-2}2^{-t})$ which corresponds to $k = 2$ in (1.1). For this case the optimal $h = h_0 = O(2^{-t/4})$ and

$$u(ih) - \bar{u}_{h,i} = O(2^{-t/2}).$$

To improve this estimate one must guarantee that $\|L_h^{-1}E_h\bar{u}_h\|_\infty << \|L_h^{-1}\|_\infty \|E_h\|_\infty \|\bar{u}_h\|_\infty$. An interesting example of such an algorithm is proposed by Babuška [2]. For Babuška's algorithm E_h has a special form, $E_h = [\epsilon_{i,i-1}, -\epsilon_{i,i-1} - \epsilon_{i,i+1}, \epsilon_{i,i+1}]$, i.e., the sum of elements of the ith row is equal to zero for $i \in (1,n)$ and $\epsilon_{i,j}$ is of order 2^{-t}. Then

$$E_h\bar{u}_h = c_1e_1 + c_ne_n + O(h2^{-t})$$

where c_1 and c_n are of order 2^{-t}, $e_1 = [1,0,\ldots,0]^T$ and $e_n = [0,\ldots,0,1]^T$. Since $\|L_h^{-1}e_i\|_\infty = O(1)$ for $i = 1$ and n, we get

$$\|u_h - \bar{u}_h\|_\infty = O(h^{-1}2^{-t}).$$

This corresponds to $k = 1$ in (1.1). The optimal $h = h_0 = O(2^{-t/3})$ and

$$u(ih) - \bar{u}_{h,i} = O(2^{-2/3t}).$$

This means that using Gaussian elimination one can compute a t/2 digit approximation to the solution of an elliptic problem whereas Babuška's algorithm yields a $\frac{2}{3}t$ digit approximation provided these two algorithms use t digit arithmetic. ∎

The aim of this paper is to study the question:

Do there exist algorithms that compute t digit approximations to the exact solutions of continuous problems using t digit arithmetic? Or stated technically: do there exist algorithms for which $k = 0$ in (1.1)?. Examples 1.1 and 1.2 indicate that if such algorithms exist, they must be specially designed to make use of some properties of the continuous problem.

We present such algorithms for continuous problems which generalize the problems described in Examples 1.1 and 1.2. We stress that the costs of these algorithms are comparable to the costs of the commonly used algorithms.

The algorithms presented in this paper utilize one or more of the following three ingredients:

(i) a property of floating point subtraction of nearby numbers,

(ii) a special algorithms for summation of very many terms,

(iii) iterative refinement of linear systems in single precision with a special algorithm for the computation of residual vectors.

Section 2 deals with these three ingredients. In Sections 3 and 4 we present algorithms for integration and for elliptic equations.

Although we only analyze the integration and elliptic equation problems in this paper, we have obtained corresponding algorithms for a number of other continuous problems such as biharmonic, parabolic and hyperbolic equations.
As in this paper these algorithms preserve certain essential properties of the continuous problems.

2. Preliminaries.

In this section we present the basic ingredients needed to construct algorithms with $O(2^{-t})$ accuracy for the approximate solutions of certain continuous problems.

(i) The first ingredient is a property of floating point subtraction of nearby numbers. Let $f\ell$ be t digit floating point binary arithmetic. Let the arithmetic register of $f\ell$ have one guard digit. We assume that for real numbers a and b that are exactly represented in $f\ell$, i.e., $a = rd(a)$ and $b = rd(b)$, we have

$$f\ell(a \square b) = (a \square b)(1+\varepsilon), \qquad |\varepsilon| \leq 2^{-t}$$

where $\square$ stands for $+,-,*$ or $/$. We additionally assume that $f\ell(a-b) = -f\ell(b-a)$.

Lemma 2.1: If $a = rd(a) \geq 1$, $b = rd(b) \geq 1$ and $|a - b| \leq 1/2$ then

$$f\ell(a-b) = a - b. \tag{2.1}$$

■

Proof: If $a = b$ then (2.1) holds trivially. Assume first that $a > b$. Then $|a - b| \leq 1/2$ reads

$$2^{c_a} m_a - 2^{c_b} m_b \leq 1/2$$

where c_a, c_b are the exponent parts of a, b and m_a, m_b are the mantissas of a,b in $f\ell$, $1/2 \leq |m_a|, |m_b| < 1$. Thus

$$2^{c_a - c_b} \leq \frac{m_b + 2^{-(1+c_b)}}{m_a} < 2^2.$$

Since $b \geq 1$ then $c_b \geq 1$. Hence $c_a = c_b + 1$ or $c_a = c_b$. Suppose that $c_a = c_b + 1$. Then subtraction is executed using the formula

$$a - b = 2^{c_a}(m_a - \frac{m_b}{2}).$$

The mantissa m_b is shifted one place to the right. The exact value of $m_a - \frac{m_b}{2}$ has at most t + 1 bits. Due to $a - b \leq 1/2$ and $a \geq 1$ we have $c_a \geq 1$ and

$$m_a - \frac{m_b}{2} \leq \frac{1}{2}\frac{1}{2^{c_a}} \leq \frac{1}{4}.$$

Thus the first bit of $m_a - \frac{m_b}{2}$ is zero. The mantissa of a - b is the normalized value of $m_a - \frac{m_b}{2}$. Thus $m_a - \frac{m_b}{2}$ is shifted at least one place to the left. The exact value of $m_a - \frac{m_b}{2}$ is stored using t bits and therefore $f\ell(a-b) = a - b$.

If $c_a = c_b$ then $m_a - m_b$ is executed. Since $m_a - m_b$ has t mantissa bits, it is exactly done in $f\ell$ and (2.1) holds.

Due to the assumption $f\ell(a-b) = -f\ell(b-a)$, the case $a < b$ is equivalent to the previous one. Hence Lemma 2.1 is proven. ∎

The essence of Lemma 2.1 is that subtraction of two nearly floating point numbers a,b which are not small is exactly performed in floating point arithmetic. This will be used in the later sections with a and b representing the values of a continuous function at nearby points.

(ii) In this subsection we present a special algorithm for summation of n terms, see [3]. This algorithm is based on the repetitive use of the Gill-Møller (GM) algorithm and will be denoted by the RGM algorithm.

To present the RGM algorithm we first recall the GM algorithm. To compute $\Sigma_{i=1}^{n} a_i$ proceed as follows, see [5], [7]:

$P_0 := S_0 := 0;$

<u>for</u> i := 1 <u>step</u> 1 <u>until</u> n <u>do</u>

<u>begin</u>

$S_i := S_{i-1} + a_i;$

$P_i := P_{i-1} + (a_i - (S_i - S_{i-1}))$

<u>end</u>;

$S_n := S_n + P_n;$

We denote $GM(n;a_1,a_2,\ldots,a_n) = S_n$.

To compute $\Sigma_{i=1}^{n} a_i$ by the RGM algorithm in t digit $f\ell$ we proceed as follows, see [3]:

For given n and t choose an integer r such that

$$n^{2/r}\, 2^{-t} \le 0.1 \quad \text{and} \quad 2.1\, r\, 2^{-t} \le 0.1. \tag{2.2}$$

Let $m = \lceil n^{1/r} \rceil$ and $a[0,i] := a_i$ for $i = 1,2,\ldots,n$ and $a[0,i] := 0$ for $i = n+1,n+2,\ldots,m^r$. Compute

<u>for</u> j := 1 <u>step</u> 1 <u>until</u> r <u>do</u>

<u>for</u> i := 1 <u>step</u> 1 <u>until</u> m^{r-j} <u>do</u>

a[j,i] := GM(m;a[j-1,(i-1)m+1],...,a[j-1,(i-1)m+m]).

Denote $RGM(r;a_1,a_2,\ldots,a_n) = a[r,1]$. Then $a[r,1]$ is computed in time proportional to n and

$$a[r,1] = \Sigma_{i=1}^{n} a_i(1+\varepsilon_i), \qquad |\varepsilon_i| \le 2.23\, r\, 2^{-t}. \tag{2.3}$$

As an example observe that for $r = 1$, the RGM algorithm coincides with the GM algorithm. For $n = 2^{at}$ one can, for large t, set $r = \lceil 3a \rceil$ and the RGM algorithm yields the exact sum of slightly perturbed terms $a_i(1+\varepsilon_i)$ with a uniform bound on ε_i given by $2.23 \lceil 3a \rceil 2^{-t}$.

(iii) In this subsection we recall iterative refinement and some of its properties as analyzed in [4]. For a nonsingular $n \times n$ matrix A consider the linear systems $Ax = b$ for different $n \times 1$ vectors b. Suppose one has an algorithm

φ that for every b finds an approximation y to $\alpha = A^{-1}b$ in t digit $f\ell$ such that

$$\|y - \alpha\| \le q\|\alpha\|$$

for some q, $q \in [0,1)$. To improve accuracy of y we apply iterative refinement as follows:

For $m := 1,2,\ldots$

-compute the residual $r^{(m)} := Ay^{(m)} - b$, $(y^{(1)} = y)$,

-solve $Ad^{(m)} = r^{(m)}$ using algorithm φ,

-compute the new approximation $y^{(m+1)} := y^{(m)} - d^{(m)}$.

Assume that the computed residual $r^{(m)}$ is of the form

$$r^{(m)} = (I + \delta I^{(m)})(Ay^{(m)} + \delta y^{(m)} - b) \tag{2.4}$$

where

$$\begin{aligned} &\|\delta I^{(m)}\| \le c_1 2^{-t}, \\ &\|\delta y^{(m)}\| \le (\eta\|y^{(m)}\| + c_2\|y^{(m)}-\alpha\|)2^{-t}\|A\|, \end{aligned} \tag{2.5}$$

for some constants c_1, c_2 and η. Here $\|\cdot\|$ denotes some norm. A slight change of the proof of Theorem 3.1 in [4] yields

<u>Theorem 2.1:</u> Let

$$\sigma_1 = (1+q)(1+2^{-t})(c_1+(1+c_1 2^{-t})(\eta+c_2))2^{-t}\text{cond}(A)+q+(2+q)2^{-t},$$

$$\sigma_2 = (1+q)(1+2^{-t})(1+c_1 2^{-t})\eta\ \text{cond}(A) + 1.$$

If $\sigma_1 < 1$ then

$$\|y^{(m+1)}-\alpha\| \leq \sigma_1^m q\|\alpha\| + (1-\sigma_1)^{-1}\sigma_2 2^{-t}\|\alpha\|.$$ ■

As usual, $\mathrm{cond}(A) = \|A\|\|A^{-1}\|$.

From Theorem 2.1 we get

Theorem 2.2: Let $\sigma_1 < 1$ and

$$k = \max\{0, \lceil \ln((1-\sigma_1)^{-1}\sigma_2 2^{-t} q^{-1})/\ln \sigma_1 \rceil\}.$$

Then

$$\|y^{(k+1)}-\alpha\| \leq \frac{2\sigma_2}{1-\sigma_1} 2^{-t}\|\alpha\|.$$ ■

Observe that σ_2 is of order $\eta\,\mathrm{cond}(A)$ and if $2^{-t}\mathrm{cond}(A)$ is much less than q, then σ_1 is of order q. In this case we have

$$k \cong \max\{0, \lceil \ln(2^{-t}\ \eta\ \mathrm{cond}(A)/(1-q)^{-1}q^{-1})/\ln q \rceil\}$$

and

$$(2.6) \qquad \|y^{k+1}-\alpha\| \leq c_3 \frac{\eta\ \mathrm{cond}(A)}{1-q} 2^{-t}\|\alpha\|$$

where c_3 is of order unity.

Thus if q is not too close to unity and one can guarantee that η is of order $1/\mathrm{cond}(A)$, then algorithm φ with iterative refinement yields an approximation with relative error of order 2^{-t}. We stress that to guarantee η to be of order $1/\mathrm{cond}(A)$, higher precision has to usually be used for the computation of the residuals $r^{(m)}$. As we shall see later, for special linear systems

$$(4.10) \qquad \lim_{\max(h,h^{-2}2^{-t})\to 0} q(h,t) = 0.$$

Thus, for small h and $h^{-2}2^{-t}$, $q(h,t)$ is small and the computed vector y is a good approximation to the vector α.

Note that (4.10) holds if φ computes y such that $\|y-\bar{\alpha}\|_\infty = O(h^{-2}2^{-t}\|\bar{\alpha}\|_\infty)$ where $\bar{\alpha} = \bar{L}_h^{-1}b$. Since $\|\alpha-\bar{\alpha}\|_\infty = O(h^{-2}2^{-t}\|\bar{\alpha}\|_\infty)$, we have $\|y-\alpha\|_\infty \le \|y-\bar{\alpha}\|_\infty + \|\alpha-\bar{\alpha}\|_\infty = O(h^{-2}2^{-t}\|\alpha\|_\infty)$ as claimed.

Observe that Gaussian elimination satisfies (4.10). Indeed, Gaussian elimination computes y which is the exact solution of $(\bar{L}_h-E_h)y = b$ where $\|E_h\|_\infty \le d_2 2^{-t}\|\bar{L}_h\|_\infty$ with d_2 of order unity. Assuming that $d_3 = d_2\|\bar{L}_h\|_\infty\|\bar{L}_h^{-1}\|_\infty 2^{-t} < 1$, we have

$$\|y-\bar{\alpha}\|_\infty \le \frac{d_3}{1-d_3}\|\bar{\alpha}\|_\infty, \qquad \bar{\alpha} = \bar{L}_h^{-1}b.$$

Since $d_3 = \Theta(h^{-2}2^{-t})$, this yields (4.10). Of course, there are many other algorithms for which (4.10) also holds. Examples include Babuška's algorithm and some iterative algorithms.

To improve the estimate (4.9) we apply iterative refinement as described in (iii) of Section 2. The computation of the residuals $r = L_h y - b$, $b = h^2 f + g$, is done in single precision by a special algorithm. We now define this algorithm. Let

$$(4.11) \qquad \delta a_i = a_{i+1} - a_i.$$

Due to (4.3), $\delta a_i = hk'(x_i) + O(h^3)$. We assume that δa_i is computed in t digit $f\ell$ such that

$$\overline{\delta a}_i = f\ell(\delta a_i) = \delta a_i + O(h2^{-t} + h^3). \tag{4.12}$$

Note that (4.12) holds if one can compute $k'(x_i)$ with the absolute error of order 2^{-t}. Then we can set $\overline{\delta a}_i := hk'(x_i)$ and

$$\overline{\delta a}_i = f\ell(hk'(x_i)) = hk'(x_i) + O(h2^{-t}) = \delta a_i + O(h2^{-t} + h^3).$$

Observe that the i-th component of $r = L_n y - b$, $y = [y_1, y_2, \dots, y_n]^T$ is given by

$$r_i = a_i(y_i - y_{i-1}) - a_{i+1}(y_{i+1} - y_i) - h^2 f_i \tag{4.13}$$

with $y_0 = g_0, y_{n+1} = g_1$. We transform (4.13) to the form

$$r_i = a_i[(y_i - y_{i-n}) - (y_{i+1} - y_i)] - \delta a_i(y_{i+1} - y_i) - h^2 f_i. \tag{4.14}$$

This is the formula from which the residual vectors will be computed. We stress that the order of arithmetic operations in (4.14) is crucial. That is, r_i should be performed as follows:

$$\begin{aligned} z_1 &:= y_i - y_{i-1}, \\ z_2 &:= y_{i+1} - y_i, \\ r_i &:= a_i * (z_1 - z_2) - \delta a_i * z_2 - h^2 * f_i. \end{aligned} \tag{4.15}$$

See [1] and [8] where a similar idea for computing r_i has been suggested.

We now show that the algorithm (4.15) computes $\bar{r}_i = f\ell(r_i)$ with a surprisingly small error.

<u>Lemma 4.1</u>: Let $y_i \geq 1$ and $|y_{i+1}-y_i| \leq 1/2$. Then

(4.16) $$\bar{r}_i - r_i = O(h^4+h^2 2^{-t} + \|y-v\|_\infty (2^{-t}+h^3))$$

and the constant in the O notation does not depend on y. ∎

<u>Proof</u>: Observe that due to Lemma 2.1, z_1 and z_2 are computed exactly in $f\ell$. Thus

$$\bar{r}_i = \bar{a}_i(z_1-z_2)(1+\varepsilon_1) - \overline{\delta a}_i z_2(1+\varepsilon_2) - h^2\bar{f}_i(1+\varepsilon_3)$$

where $\bar{a}_i$, $\overline{\delta a}_i$, $\bar{f}_i$ are given by (4.6), (4.12) and $\varepsilon_i = O(2^{-t})$. From (4.6) and (4.12) we get

$$\bar{r}_i - r_i = O(|z_1-z_2|2^{-t} + |\delta a_i z_2|2^{-t} + h|z_2|2^{-t}$$
$$+ h^3|z_2| + h^2 2^{-t}).$$

Note that $\delta a_i = O(h)$ and

$$z_2 = v_{i+1} - v_i + (y_{i+1}-v_{i+1}) - (y_i-v_i) = O(h+\|y-v\|_\infty)$$

due to (4.5). Similarly

$$z_1-z_2 = 2v_i-v_{i-1}-v_{i+1} + 2(y_i-v_i) - (y_{i-1}-v_{i-1})$$
$$- (y_{i+1}-v_{i+1}) = O(h^2+\|y-v\|_\infty).$$

Hence

$$\bar{r}_i - r_i = O(h^2 2^{-t} + \|y-v\|_\infty (2^{-t}+h^3) + h^4).$$

Since none of the constants appearing in the O notation depend on y, (4.16) holds. ∎

We are ready to prove the main theorem of this paper.

Theorem 4.1: Let (4.6) and (4.12) hold. Then an algorithm φ satisfying (4.9) and (4.10) with $k = O(\ln \frac{1}{h})$ iterative refinement steps using the algorithm (4.15) computes the vector $\bar{u}_h = [\bar{u}_1, \bar{u}_2, \ldots, \bar{u}_n]^T$ such that

$$(4.17)\qquad \begin{aligned} v_i - \bar{u}_i &= O(h^2), \\ u(x_i) - \bar{u}_i &= O(h^2), \end{aligned}$$

whenever $h^{-2}2^{-t}$ is of order unity, i.e., there exist positive constants d_4, d_5 and d_6 such that $h \le d_4$, $h^{-2}2^{-t} \le d_5$ imply $|v_i - \bar{u}_i| \le d_6 h^2$ and $|u(x_i) - \bar{u}_i| \le d_6 h^2$.

If the cose of φ is proportional to h^{-1} then for $h = O(2^{-t/2})$, $\bar{u}_h$ is computed in time proportional to $t2^{t/2}$ and

$$(4.18)\qquad u(x_i) - \bar{u}_i = O(2^{-t}).$$ ∎

Proof: We use Theorem 2.1 with the infinity norm to show (4.17). The algorithm φ computes the vector y such that

$$\|y-v\|_\infty \leq q\|v\|_\infty, \quad q = q(h,t),$$

see (4.9) and (4.10). For small h and $h^{-2}2^{-t}$, (4.5), (4.10) and $u(x_i) \geq 2$ yield that the y_i are close to the $u(x_i)$ and $y_{i+1} - y_i$ are close to zero. Hence $y_i \geq 1$, $|y_{i+1}-y_i| \leq \frac{1}{2}$ and we can apply Lemma 4.1 for the computed vector $\bar{r}_1 = f\ell(L_h y-b)$. Due to (4.16), (2.4) and (2.5) hold with $c_1 = 0$, $\eta = O(h^4 2^t + h^2)$ and $c_2 = O(1 + h^3 2^t)$. The parameters σ_1 and σ_2 of Theorem 2.1 satisfy the relations,

$$\sigma_1 = O(h^{-2}2^{-t} + h + q), \quad \sigma_2 = O(h^2 2^t + 1).$$

For small h and $h^{-2}2^{-t}$, σ_1 is small. This means that the speed of convergence of iterative refinement is fast. For small h, all components of $y^{(1)}$ as well as $y^{(m)}$ are close to two and $y_{i+1}^{(m)} - y_i^{(m)}$ are close to zero. Hence Lemma 4.1 can be applied for any m. After k steps where $k = O(\ell n \frac{1}{h})$, we have due to Theorem 2.2

$$\|y^{(k+1)}-v\|_\infty = O(\sigma_2 2^{-t}) = O(h^2).$$

Setting $\bar{u}_h = y^{(k+1)}$ and using (4.5), (4.16), we obtain (4.17).

To show (4.18), observe that the cost of computing $\bar{u}_h$ is proportional to $\frac{1}{h}\,\ell n\,\frac{1}{h}$. For $h = O(2^{-t/2})$, it is proportional to $t2^{t/2}$ as claimed. ∎

<u>Remark 4.1</u>: Suppose that (4.12) is slightly strengthened.

That is, let $\overline{\delta a_i} = f\ell(\delta a_i) = \delta a_i + O(h2^{-t})$. This holds, for instance, if $k(x) \equiv$ const in (4.1) which implies $\delta a_i \equiv 0$. Then the proof of Theorem 4.1 yields that

$$\|v-\bar{u}_h\|_\infty = O(2^{-t}),$$

i.e., we can solve the linear system (4.4) whose condition number is of order h^{-2} using only single precision with accuracy independent of h. ∎

We now briefly indicate how to generalize our analysis to the multidimensional elliptic equations of the form

$$(4.18)\qquad \begin{aligned} -\Sigma_{j=1}^{m} \frac{\partial}{\partial x_j}\left(k_j(x)\frac{\partial u}{\partial x_j}\right)(x) &= f(x), && x \in D,\\ u(x) &= g(x), && x \in \partial D, \end{aligned}$$

where $D = (0,1)^m$, $k_j(x) \geq k_j > 0$ for smooth functions k_j, f and g. As we mentioned before we can assume without loss of generality that $u(x) \geq 2$ for $x \in \bar{D}$.

For $x = [i_1h, i_2h, \ldots, i_mh]^T$, $1 \leq i_j \leq N$, $h = 1/(N+1)$, we approximate (4.18) in each direction as in (4.2). We obtain the following difference scheme

$$\Lambda_h v(x) = \frac{1}{h^2}\, \Sigma_{j=1}^{m}[a_j(x)(v(x)-v(x-he_j)) - a_j(x+he_j)(v(x+he_j) - v(x))]$$

where $e_j = [0,\ldots,\underset{j}{1},\ldots,0]^T$ and

$$\frac{a_j(x+he_j) - a_j(x)}{h} = \frac{\partial k_j}{\partial x_j}(x) + O(h^2),$$

$$\frac{a_j(x+he_j) + a_j(x)}{2} = k_j(x) + O(h^2).$$

This difference scheme is equivalent to the $n \times n$ linear system $Av = b$ whose form is similar to (4.4) with $n = N^m = (\frac{1}{h} - 1)^m$. The condition number of this system is $\Theta(h^{-2})$.

We assume that $a_j(x)$, $f(x)$ and $g(x)$ for the meshpoints x are computed with absolute error of order 2^{-t}, see (4.6). As in (4.12) assume that $\delta a_j(x) = a_j(x+he_j) - a_j(x)$ is computed with absolute error of order $h2^{-t} + h^3$.

Let φ be an arbitrary algorithm solving $Av = b$ which produces in t digit $f\ell$ a vector y satisfying (4.9) and (4.10).

Following the proof of Theorem 4.1 one can obtain

<u>Theorem 4.2</u>: The algorithm φ satisfying (4.9) and (4.10) with $k = O(\ell n \frac{1}{h})$ iterative refinement steps using the algorithm (4.15) in each direction computes the vector $\bar{u}_h$ such that

$$\|\bar{u}_h - v\|_\infty = O(h^2),$$

$$u(x) - \bar{u}_h(x) = O(h^2)$$

whenever $h^{-2}2^{-t}$ is of order unity; x is a meshpoint and $\bar{u}_h(x)$ is the corresponding component of $\bar{u}_h$. ∎

We comment on the assumption (4.10). As already observed, this holds for many algorithms for the one dimensional case, $m = 1$. For $m \geq 2$, many efficient direct algorithms compute y such that

$$\|y-v\|_\infty \leq d_7 h^{-2} 2^{-t} \|v\|_\infty \tag{4.19}$$

where d_7 depends on n. For instance, for algorithms using the Fast Fourier transforms, $d_7 = \Theta(\ell n\ n)$, see [6] and [9]. We stress that d_7 has to be of order unity if (4.10) is satisfied. We know no direct algirthms for which (4.19) holds with $d_7 = O(1)$ for $m \geq 2$. We doubt if such algorithms exist.

For $m \leq 3$, there exists an iterative algorithm for which (4.19) holds with $d_7 = O(1)$. This is Chebyshev's algorithm. To show this, recall that Chebyshev's algorithm approximates the solution v of $Av = b$ in the spectral norm $\|\cdot\|_2$. From Lemma 4.1 it is easy to observe that $\bar{r} = f\ell(Ay - b)$ satisfies

$$\|\bar{r}-r\|_2 = O(h^4\|y\|_2 + (2^{-t}+h)\|y-v\|_2) \tag{4.20}$$

whenever $h^{-2}2^{-t}$ is of order unity. Theorem 5.1 of [12] yields that Chebyshev's algorithm with the algorithm (4.15) for computation of residuals produces a vector z such that

$$\|z-v\|_2 \leq d_8 h^{-2} 2^{-t} \|v\|_2 \tag{4.21}$$

with d_8 of order unity. Applying iterative refinement to Chebyshev's algorithm. Theorem 2.1 and (4.20) yield that we can compute a vector y such that

$$(4.22) \qquad \|y-v\|_2 \le d_9 h^2 \|v\|_2$$

with d_9 of order unity. From (4.21) we have in the infinity norm

$$\|y-v\|_\infty \le d_9 h^2 \sqrt{n}\ \|v\|_\infty \le d_9 h^{2-m/2} \|v\|_\infty .$$

Thus $q(h,t) = d_9 h^{2-m/2}$ and (4.10) is satisfied since $2-m/2 > 0$.

It is easy to see why the assumption $m \le 3$ is needed for iterative algorithms which approximate the solution in the spectral norm. Even if such an algorithm computes an approximation y with full precision in t digit $f\ell$, $\|y-v\|_2 = O(2^{-t}\|v\|_2)$ and $\|v\|_2 = \Theta(\sqrt{n}\|v\|_\infty)$, then $\|y-v\|_\infty = O(2^{-t}\|v\|_\infty\sqrt{n})$. Since $\sqrt{n} \cong h^{-m/2}$ and h^2 can be of order 2^{-t}, then

$$\|y-v\|_\infty = O(h^{2-m/2}\|v\|_\infty).$$

Thus y approximates v in the infinity norm with some precision whenever $2 - m/2$ is positive, i.e., $m \le 3$.

For $m \ge 4$, we know no algorithms for which (4.10) is satisfied, i.e., the problem of designing algorithms which approximate $u(x_i)$ with order 2^{-t} using t digit $f\ell$ is open.

Acknowledgements

We are grateful to A. Kiełbasiński and J.F. Traub for their valuable comments and suggestions.

References

[1] Axelsson, O., Gustafsson, I.: A preconditioned conjugate gradient method for finite element equations, which is stable for rounding errors. Inf. Proc. 80, pp. 723-728.

[2] Babuška, J.: Numerical stability in problems of linear algebra. SIAM J. Num. Anal., 1972, 9, pp. 53-77.

[3] Jankowski, M., Smoktunowicz, A., Woźniakowski, H.: A note on floating-point summation of very many terms. Journal of Information Processing and Cybernetics--EIK, to appear.

[4] Jankowski, M., Woźniakowski, H.: Iterative refinement implies numerical stability. BIT, 1977, 17, pp. 303-311.

[5] Kiełbasiński, A.: Summation algorithm with corrections and some of its applications, Mat. Stos., 1973, I, pp. 22-41 (in Polish).

[6] Mejran,Z.: Algorithm for solving linear algebraic systems arising from the difference method for Poisson's equation. Mat. Stos., 1978, XII, pp. 35-41 (in Polish).

[7] Møller, O.: Quasi-double-precision in floating-point addition. BIT, 1965, 5, pp. 37-50, 251-255.

[8] Pohl, P.: Iterative Improvement Without Double Precision in a Boundary Value Problem. BIT, 1974, 14, pp. 361-365.

[9] Ramos, G.: Roundoff error analysis of the fast Fourier transform. Tech. Rep. STAN-CS-70-146, 1970, Stanford University.

[10] Samarskij, A.A.: Theory of the difference schemes. Moscow 1977 (in Russian).

[11] Wilkinson, J.H.: Rounding errors in algebraic processes. Englewood Cliffs, N.J., Prentice-Hall 1963.

[12] Woźniakowski, H.: Numerical Stability of the Chebyshev Method for the Solution of Large Linear Systems. Numer. Math., 1977, 28, pp. 191-209.

that arise from the discretization of certain continuous problems it is possible to compute the residuals in <u>single</u> precision such that η is of order 1/cond(A) although cond(A) is huge. In this case we guarantee $O(2^{-t})$ precision of the computed approximation while still performing all operations in single precision.

3. Integration.

Let $f: [0,1] \to \mathbb{R}$ belong to the class $C_{s,\lambda}$, i.e., f is s-times differentiable function, $s \geq 0$, and its s-th derivative satisfies a Hölder condition of order λ, $\lambda \in [0,1]$, i.e., $|f^{(s)}(x) - f^{(s)}(y)| \leq M|x-y|^{\lambda}$, $\forall x,y \in [0,1]$, for some constant M. We assume that $s + \lambda > 0$. We wish to approximate

$$u = S(f) = \int_0^1 f(t)dt.$$

For $h = 1/n$ consider a quadrature formula of the form

$$u_h = S_h(f) = \Sigma_{i=1}^{n} A_i f(x_i)$$

where A_i and x_i depend on h. We assume that the weights A_i are nonnegative and the quadrature formula S_h is convergent for continuous functions. The truncation error is assume to be

$$u - u_h = O(h^p), \qquad p = \lambda + s,$$

for functions f from the class $C_{s,\lambda}$.

Assume that the weights A_i and the function values $f(x_i)$ can be computed in $f\ell$ with high relative precision, i.e.,

$$\tilde{A}_i = f\ell(A_i) = A_i(1 + \delta A_i), \qquad |\delta A_i| \leq d_1 2^{-t},$$

$$\tilde{f}_i = f\ell(f(x_i)) = f(x_i)(1 + \delta f_i), \quad |\delta f_i| \leq d_2 2^{-t}$$

for some constants d_1 and d_2. Let $a_i = f\ell(\tilde{A}_i \tilde{f}_i) = \tilde{A}_i \tilde{f}_i (1+\tilde{\delta}_i)$,

$|\tilde{\delta}_i| \leq 2^{-t}$. We compute $\Sigma_{i=1}^{n} a_i$ by the RGM algorithm with $r = \lceil 2/p \rceil$. Assume that t satisfies

(3.2) $$2.1 \lceil 2/p \rceil 2^{-t} \leq 0,1.$$

For $h \geq 10^{r/2} 2^{-t/p}$, (2.2) holds and the RGM produces

(3.3) $$\bar{u}_h = RGM(r; a_1, a_2, \ldots, a_n) = \Sigma_{i=1}^{n} A_i f(x_i)(1+\delta_i)$$

where $1 + \delta_i = (1+\delta A_i)(1+\delta f_i)(1+\tilde{\delta}_i)(1+'\varepsilon_i)$ with $|'\varepsilon_i| \leq 2.23r2^{-t}$ due to (2.3). Thus

$$|\delta_i| \leq (d_1+d_2+1+2.23r)2^{-t} + O(2^{-2t}).$$

Note that $|u_h - \bar{u}_h| \leq (\Sigma_{i=1}^{n} A_i |f(x_i)|) \max_{1 \leq i \leq n} (|\delta_i|)$. Since $\Sigma_{i=1}^{n} A_i |f(x_i)|$ converges to $\int_0^1 |f(t)| dt$ we have

$$u_h - \bar{u}_h = O(2^{-t}).$$

Hence we have proven

Theorem 3.1: Let (3.1) and (3.2) hold. Then the RGM algorithm with $r = \lceil 2/p \rceil$ computes $\bar{u}_h$ in time proporitional to h^{-1} such that

$$\bar{u}_h - \int_0^1 f(t)dt = O(h^p)$$

whenever $h \geq 10^{r/2} 2^{-t/p}$. For h close to $10^{r/2} 2^{-t/p}$,

$$\bar{u}_h - \int_0^1 f(t)dt = O(2^{-t}). \quad ∎$$

Observe that for smooth problems, i.e., for functions f for which $p \geq 2$, we have $r = 1$ and the RGM algorithm coincides with the GM algorithm. For $p \in [1,2)$, we get $r = 2$. For nonsmooth problems, i.e., for functions f for which $s = 0$, we have $p = \lambda$ and $r = \lceil 2/\lambda \rceil$. Thus for small λ, r is large. Even in this case we can (theoretically) find a t digit approximation to the integral of f although the cost of the RGM algorithm is unreasonably high, since it is proporition to $2^{t/\lambda}$.

4. Elliptic Equations.

In this section we first analyze a one dimensional elliptic equation in detail and next indicate how our results can be generalized for the multidimensional case.

Consider the elliptic equation

$$(4.1)\qquad \begin{aligned} &-\frac{d}{dx}\left(k(x)\frac{du}{dx}\right)(x) = f(x), \qquad x \in (0,1), \\ &u(0) = g_0, \qquad u(1) = g_1. \end{aligned}$$

We assume that $0 < k_0 \leq k(x)$. For simplicity we also assume that k and f are sufficiently smooth.

We wish to find $v_i = v_h(x_i)$ which approximates $u(x_i)$ for $x_i = ih$ with the meshsize $h = 1/(n+1)$. To find v_i, (4.1) is discretized as follows. The operator $-\frac{d}{dx}(k(x)\frac{du}{dx})$ is approximated with error of order h^2 by the operator Λ_h, see [10, pp. 149-170],

$$(4.2)\qquad \Lambda_h v_i = \frac{1}{h^2}[a_i(v_i - v_{i-1}) - a_{i+1}(v_{i+1} - v_i)]$$

where a_i satisfies two conditions

$$(4.3)\qquad \begin{aligned} &\frac{a_{i+1} - a_i}{h} = k'(x_i) + O(h^2). \\ &\frac{a_{i+1} + a_i}{2} = k(x_i) + O(h^2). \end{aligned}$$

For instance, a_i can be equal to $k(x_i - h/2)$ or

$(k(x_i) + k(x_i - h))/2$.

Let $f_i = f(x_i)$. From (4.2) we get the three point difference scheme $\Lambda_h v_i = f_i$ which is equivalent to the $n \times n$ system of linear equations

$$(4.4) \qquad L_h v = h^2 f + g$$

where L_h is a $n \times n$ symmetric positive definite tridiagonal matrix with the i-th row given by $[0,\ldots,-a_i, a_{i+1}+a_i, -a_{i+1},\ldots,0]$, $v = [v_1, v_2, \ldots, v_n]^T$, $f = [f_1, f_2, \ldots, f_n]^T$ and $g = [a_1 g_0, 0, \ldots, 0, a_{n+1} g_1]^T$. It is well known that

$$(4.5) \qquad u(x_i) - v_i = O(h^2)$$

and that $\text{cond}(L_h) = \|L_h\|_\infty \|L_h^{-1}\|_\infty = O(h^{-2})$.

Observe that without loss of generality we can assume that $u(x) \geq 2$ for $x \in [0,1]$. Indeed, it is enough to replace $f(x)$ and g_0, g_1 with, for example $\tilde{f}(x) = f(x) + d_1$, $\tilde{g}_0 = g_0 + d_1$, $\tilde{g}_1 = g_1 + d_1$. Then the solution $\tilde{u}(x) = u(x) + d_1$. Since

$$\|u\|_\infty \leq \frac{1}{k_0}\|f\|_\infty + \max(|g_0|, |g_1|) =: d_0,$$

see [10], then setting $d_1 = 2 + d_0$ we get $\tilde{u}(x) \geq d_1 - \|u\|_\infty \geq 2$ as claimed. For small h, (4.5) yields that all v_i are close to 2 and the $v_{i+1} - v_i$ are close to zero.

We now turn to the computational aspects of solving (4.4) in t digit $f\ell$. We first compute the coefficients of

L_h, f and g. We assume that the values f_i, a_i, g_0 and g_1 are computed with the absolute error of order 2^{-t}, i.e.,

$$\bar{f}_i = f\ell(f_i) = f_i + O(2^{-t}), \qquad i = 1,2,\dots,n,$$
$$(4.6) \qquad \bar{a}_i = f\ell(a_i) = a_i + O(2^{-t}), \qquad i = 1,2,\dots,n,$$
$$\bar{g}_j = f\ell(g_j) = g_j + O(2^{-t}), \qquad j = 0,1.$$

Let $\bar{L}_h$, $\bar{f}$ and $\bar{g}$ denote the computed matrix L_h and the computed vectors f and g. Thus, instead of the linear system (4.4) we have

$$(4.7) \qquad \bar{L}_h\bar{v} = h^2\bar{f} + \bar{g}.$$

It is easy to show that $\|\bar{L}_h\|_\infty\|\bar{L}_h^{-1}\|_\infty = O(h^{-2})$ and

$$(4.8) \qquad \|v-\bar{v}\|_\infty = O(h^{-2}2^{-t}).$$

Let φ be an arbitrary algorithm for solving $L_hx = b$ with an arbitrary vector b. We assume that φ satisfies the assumption of Theorem 2.1, i.e., φ produces in t digit $f\ell$ a vector y such that

$$(4.9) \qquad \|y-\alpha\|_\infty \le q\|\alpha\|_\infty, \qquad \alpha = L_h^{-1}b,$$

with $q < 1$. Observe that q is a function of the meshsize h and the mantissa length t, $q = q(h,t)$. We need to assume that