Iterative refinement: how to use it to "verify" a computed result?

Hong Diep Nguyen and Nathalie Revol
INRIA - LIP (UMR 5668 CNRS - ENS de Lyon - INRIA - UCBL) - Université de Lyon

NSV-3, 15 July 2010

Agenda

Agenda

What is iterative refinement

Compute a solution $\hat{x}$ of the problem at hand.

Due to floating-point computations and roundoff errors, $\hat{x}$ is not the exact solution x of the problem. There is an error $e = x - \hat{x}$.

For some problems, the error e is a solution of the same problem with different constants.

Iterative refinement:

1. **solve the original problem: compute $\hat{x}$**
2. **solve the problem having e as solution: compute $\hat{e}$**
3. **correct the computed solution: $x' \leftarrow \hat{x} + \hat{e}$**

Usually, x' is a more accurate solution than $\hat{x}$.

Example: summation $s = \sum_{i=1}^{n} x_i$

Let us denote by $\hat{s}_i = \mathrm{fl}(x_i + \hat{s}_{i-1})$ and $\hat{s}_1 = x_1$,
$\varepsilon_i = (x_i + \hat{s}_{i-1}) - \mathrm{fl}(x_i + \hat{s}_{i-1})$: roundoff error of the i^{th} addition.

$$\text{The total error } e = s - \hat{s} = \sum_{i=2}^{n} \varepsilon_i .$$

The error is the solution of a summation problem.

Algorithm:

1. compute $\hat{s}_i$ and ε_i using TwoSum
2. compute $\hat{e} = \sum \varepsilon_i$
3. $s' = \mathrm{fl}(\hat{s}_n + \hat{e})$

cf. Pichat (1972) and Neumaier (1974).

Example: solving a linear system $Ax = b$

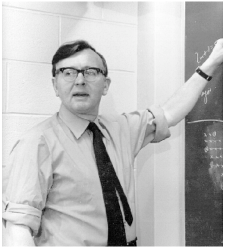
Solve $Ax = b$: computed solution $\hat{x}$.

The error $e = x - \hat{x}$ satisfies $Ae = Ax - A\hat{x} = b - A\hat{x}$.
Residual $r := b - A\hat{x}$
The error satisfies $Ae = r$.

Iterative refinement:

1. compute $\hat{x}$ approximate solution of $Ax = b$
2. compute residual $r = b - A\hat{x}$
3. compute $\hat{e}$ approximate solution of $Ae = r$
4. correct the solution: $x' = \hat{x} + \hat{e}$

(Wilkinson, 1948)



Example: the wave equation, a linear PDE

Equation:

$$\frac{\partial^2 u(x,t)}{\partial t^2} - c^2 \frac{\partial^2 u(x,t)}{\partial x^2} = s(x,t).$$

Cf. talk by Sylvie Boldo:
The error satisfies the same equation as the sought function u.

Iterative refinement could be:

1. compute $\hat{u}$ approximate solution of the wave equation
2. compute $\hat{e}$ approximate solution of the wave equation (with proper initial/boundary values)
3. correct the solution: $u' = \hat{u} + \hat{e}$

Example: solving $f(x) = 0$

Computed solution $\hat{x}$.

The error $e = x - \hat{x}$ can be deduced from

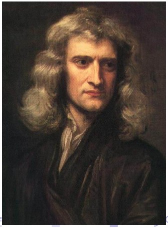
$$\begin{array}{rcccl} & f(x) & = & 0 & = & f(\hat{x}) + (x - \hat{x})f'(\hat{x}) + \mathcal{O}(e^2) \\ \Rightarrow & & & e & \simeq & -f(\hat{x})/f'(\hat{x}) \end{array}$$

The error e does not satisfy the same equation as x...

but let's use its approximation!

Iterative refinement:

1. compute $\hat{x}$ approximate solution of $f(x) = 0$
2. compute $\hat{e} = -f(\hat{x})/f'(\hat{x})$ approximation of e
3. correct the solution:
   $x' = \hat{x} + \hat{e} = \hat{x} - f(\hat{x})/f'(\hat{x})$



(Newton-Raphson, 1669-1690-1740)

Example: solving $f(x) = 0$

Computed solution $\hat{x}$.

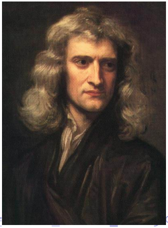
The error $e = x - \hat{x}$ can be deduced from

$$\begin{array}{rcl} f(x) & = & 0 = f(\hat{x}) + (x-\hat{x})f'(\hat{x}) + \mathcal{O}(e^2) \\ \Rightarrow \quad & e & \simeq -f(\hat{x})/f'(\hat{x}) \end{array}$$

The error e does not satisfy the same equation as x...
but let's use its approximation!

Iterative refinement:

1. compute $\hat{x}$ approximate solution of $f(x) = 0$
2. compute $\hat{e} = -f(\hat{x})/f'(\hat{x})$ approximation of e
3. correct the solution:
   $x' = \hat{x} + \hat{e} = \hat{x} - f(\hat{x})/f'(\hat{x})$



(Newton-Raphson, 1669-1690-1740)

Example: optimisation $\min f(x)$

The exact solution x satisfies $f'(x) = 0$.

Same problem as before...
Computed solution $\hat{x}$.

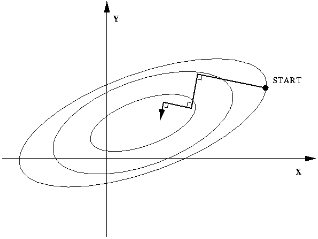
The error $e = x - \hat{x}$ is approximated by $e \simeq -f'(\hat{x})/f''(\hat{x})$.

The computed solution is corrected:

$$x' = \hat{x} + e = \hat{x} - f'(\hat{x})/f''(\hat{x})$$

.

This is the **steepest descent method.**



Agenda

What is iterative refinement

How to use iterative refinement to verify a computed result?

Influence of the computing precision

Conclusion and future work

How to use iterative refinement to "verify" a computed result?

In the computer arithmetic community, **verify** means

- establish a pen-and-paper proof (using the specifications of floating-point arithmetic. . .)
- compute an enclosure of the (unknown) exact result

but usually, no computer-proof-checked proof,

or not yet.

How to use iterative refinement to "verify" a computed result?

In the computer arithmetic community, **verify** means

- establish a pen-and-paper proof (using the specifications of floating-point arithmetic. . .)
- compute an enclosure of the (unknown) exact result

but usually, no computer-proof-checked proof,

or not yet.

How to use iterative refinement to "verify" a computed result?

In the computer arithmetic community, **verify** means

- establish a pen-and-paper proof (using the specifications of floating-point arithmetic. . .)
- compute an enclosure of the (unknown) exact result

but usually, no computer-proof-checked proof,

or not yet.

There is no such thing ~~as a free beer~~ as a bug free computation
interval arithmetic to contain the results

(Moore 1966, Kulisch 1983, Neumaier 1990, Rump 1994, Alefeld and Mayer 2000. . .)

Principle

Numbers are replaced by intervals.
π replaced by $[3.14159, 3.14160]$

For instance, the content of my wallet is between 20 and 30 £,
$\in [20, 30]$ £.

The “Thou shalt not lie” principle

Interval arithmetic computes an enclosure of the (unknown) exact result.

This is considered as **verified computation**.

Interval arithmetic in a nutshell

$$[10, 20] + [5, 10] = [15, 30]$$
$$[-2, 3] + [5, 7] = [3, 10]$$

$$[-3, 2] * [-3, 2] = [-6, 9] \text{ differs from } [-3, 2]^2 = [0, 9]$$

$$[-3, 2]/[0.5, 1] = [-6, 4]$$

$$X \diamond Y = \{x \diamond y \,/\, x \in X,\ y \in Y\}$$

$$\exp[-2, 3] = [\exp(-2), \exp(3)]$$
as exp is an increasing function.

$$\sin[\pi/3, \pi] = [0, 1]$$
beware, sin is non monotonic.

There is no such thing ~~as a free beer~~ as a bug free computation
interval arithmetic to contain the errors

Notation: interval quantities are **boldface**.

Computing an approximate sum:

1. compute $\mathbf{s_i} = x_i + \mathbf{s_{i-1}}$

$\Rightarrow$ $s \in \mathbf{s_n}$

Verifying the sum:

1. compute $\hat{s}_i = \mathrm{fl}(x_i + \hat{s}_{i-1})$ and ε_i using TwoSum
2. compute $\mathbf{e} = \sum[\varepsilon_i]$

$\Rightarrow$ $s \in \hat{s}_n + \mathbf{e}$

Width of $\mathbf{e} \simeq 2^{-53}$ width of $\mathbf{s_n}$.

Verifying the solution of $f(x) = 0$: interval Newton algorithm

Newton-Raphson:

1. compute $\hat{x}$ approximate solution of $f(x) = 0$
2. compute $\hat{e} = -f(\hat{x})/f'(\hat{x})$ approximation of e
3. correct the solution: $x' = \hat{x} + \hat{e} = \hat{x} - f(\hat{x})/f'(\hat{x})$

Interval Newton-Raphson:

1. compute $\mathbf{x}$ current iterate, enclosing the solution of $f(x) = 0$
2. choose any $\hat{x} \in \mathbf{x}$
3. compute $\mathbf{e} = -f(\hat{x})/f'(\mathbf{x})$ enclosing the error
4. correct the solution: $\mathbf{x}' = \hat{x} + \mathbf{e} = \hat{x} - f(\hat{x})/f'(\mathbf{x})$

Verifying the solution of $f(x) = 0$: interval Newton algorithm

Newton-Raphson:

1. compute $\hat{x}$ approximate solution of $f(x) = 0$
2. compute $\hat{e} = -f(\hat{x})/f'(\hat{x})$ approximation of e
3. correct the solution: $x' = \hat{x} + \hat{e} = \hat{x} - f(\hat{x})/f'(\hat{x})$

Interval Newton-Raphson:

1. compute $\mathbf{x}$ current iterate, enclosing the solution of $f(x) = 0$
2. choose any $\hat{x} \in \mathbf{x}$
3. compute $\mathbf{e} = -f(\hat{x})/f'(\mathbf{x})$ enclosing the error
4. correct the solution: $\mathbf{x}' = \hat{x} + \mathbf{e} = \hat{x} - f(\hat{x})/f'(\mathbf{x})$

Verifying the solution of $Ax = b$

Iterative refinement:

1. compute $\hat{x}$ approximate solution of $Ax = b$
2. compute residual $r = b - A\hat{x}$
3. compute $\hat{e}$ approximate solution of $Ae = r$
4. correct the solution: $x' = \hat{x} + \hat{e}$

Interval iterative refinement:

1. compute $\hat{x}$ approximate solution of $Ax = b$
2. compute residual $\mathbf{r}$ enclosing $b - A\hat{x}$
3. compute $\mathbf{e}'$, enclosing solution of $A\mathbf{e} = \mathbf{r}$
4. correct the solution: $x' = \hat{x} + \mathrm{mid}(\mathbf{e}')$, $\mathbf{e}'' = \mathbf{e}' - \mathrm{mid}(\mathbf{e}')$

Difficulty: computing $\mathbf{e}'$ is an iterative process, the determination of the starting point **which encloses the error** is not obvious.

Verifying the solution of $Ax = b$

Iterative refinement:

1. compute $\hat{x}$ approximate solution of $Ax = b$
2. compute residual $r = b - A\hat{x}$
3. compute $\hat{e}$ approximate solution of $Ae = r$
4. correct the solution: $x' = \hat{x} + \hat{e}$

Interval iterative refinement:

1. compute $\hat{x}$ approximate solution of $Ax = b$
2. compute residual $\mathbf{r}$ enclosing $b - A\hat{x}$
3. compute $\mathbf{e}'$, enclosing solution of $A\mathbf{e} = \mathbf{r}$
4. correct the solution: $x' = \hat{x} + \mathrm{mid}(\mathbf{e}')$, $\mathbf{e}'' = \mathbf{e}' - \mathrm{mid}(\mathbf{e}')$

Difficulty: computing $\mathbf{e}'$ is an iterative process, the determination of the starting point **which encloses the error** is not obvious.

Verifying the solution of $Ax = b$

Iterative refinement:

1. compute $\hat{x}$ approximate solution of $Ax = b$
2. compute residual $r = b - A\hat{x}$
3. compute $\hat{e}$ approximate solution of $Ae = r$
4. correct the solution: $x' = \hat{x} + \hat{e}$

Interval iterative refinement:

1. compute $\hat{x}$ approximate solution of $Ax = b$
2. compute residual $\mathbf{r}$ enclosing $b - A\hat{x}$
3. compute $\mathbf{e}'$, enclosing solution of $A\mathbf{e} = \mathbf{r}$
4. correct the solution: $x' = \hat{x} + \mathrm{mid}(\mathbf{e}')$, $\mathbf{e}'' = \mathbf{e}' - \mathrm{mid}(\mathbf{e}')$

Difficulty: computing $\mathbf{e}'$ is an iterative process, the determination of the starting point **which encloses the error** is not obvious.

Agenda

Influence of the computing precision

Verifying the solution of $Ax = b$

Interval iterative refinement:

1. compute $\hat{x}$ approximate solution of $Ax = b$
2. compute residual $\mathbf{r}$ enclosing $b - A\hat{x}$
3. compute $\mathbf{e}'$, enclosing solution of $A\mathbf{e} = \mathbf{r}$
4. correct the solution: $x' = \hat{x} + \mathrm{mid}(\mathbf{e}')$, $\mathbf{e}'' = \mathbf{e}' - \mathrm{mid}(\mathbf{e}')$

**Residual $r = b - A\hat{x}$ is subject to cancellation,
it should be computed in higher precision.**

Mixed precision iterative refinement for $Ax = b$

Mixed precision (interval) iterative refinement:

1. compute $\hat{x}$ approximate solution of $Ax = b$
2. compute residual $\mathbf{r}$ enclosing $b - A\hat{x}$ **in higher precision**
3. compute $\mathbf{e}'$, enclosing solution of $A\mathbf{e} = \mathbf{r}$
4. correct the solution: $x' = \hat{x} + \mathrm{mid}(\mathbf{e}')$, $\mathbf{e}'' = \mathbf{e}' - \mathrm{mid}(\mathbf{e}')$

Modified mixed precision iterative refinement for $Ax = b$

Proposal: compute also $\hat{x}$ in extended precision.

Modified mixed precision interval iterative refinement:

1. compute $\hat{x}$ approximate solution of $Ax = b$ **in higher precision**
2. compute residual $\mathbf{r}$ enclosing $b - A\hat{x}$ **in higher precision**
3. compute $\mathbf{e}'$, enclosing solution of $A\mathbf{e} = \mathbf{r}$
4. correct the solution: $x' = \hat{x} + \mathrm{mid}(\mathbf{e}')$, $\mathbf{e}'' = \mathbf{e}' - \mathrm{mid}(\mathbf{e}')$ **in higher precision**.

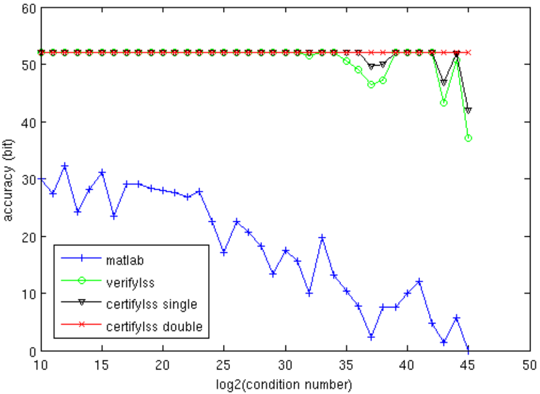
inspired from **Langou, Langou, Luszczek, Kurzak, Buttari and Dongarra (2006)** and **Demmel, Hida, Kahan, Li, Mukherjee and Riedy (ACM TOMS 32(2), 2006).**

Modified mixed precision iterative refinement for $Ax = b$: experimental results

Comparison between:

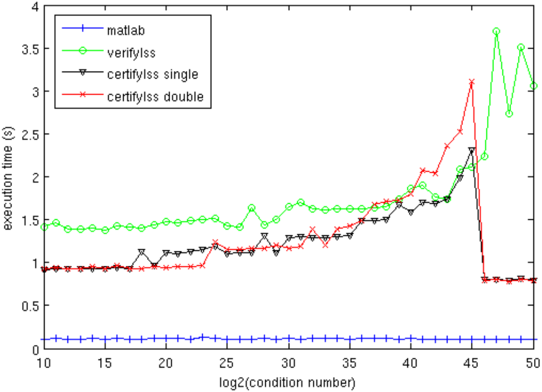
- MatLab $x = A \ b$ (non verified)
- `verifylss`: certified implementation by Rump, in IntLab
- `certifylss single`: residual computed using twice the computing precision, solution computed using the computing precision
- `certifylss double`: residual computed using twice the computing precision, solution computed using twice the computing precision

Modified mixed precision iterative refinement for $Ax = b$: experimental results



Precision of the solution in function of the condition number of A.

Modified mixed precision iterative refinement for $Ax = b$: experimental results



Computing time in function of the condition number of A.

Increasing the computing precision for x theoretical results

Let us iterate this refinement: $x_{i+1} = x_i + \mathrm{mid}(\mathbf{e_i})$.

Following **Higham**, one can prove that:

$$|x - x_{i+1}| = G_i |x - x_i| + g_i$$

where
$G_i \leq 2^{-53} \cdot |A^{-1}| \cdot W + 2 \cdot 2^{-53} \cdot (I + 2^{-53} \cdot |A^{-1}| \cdot W) \cdot |A^{-1}| \cdot |A|$
and
$g_i \leq 2n \cdot 2^{-106}(I + 2^{-53} \cdot |A^{-1}| \cdot W) \cdot |A^{-1}| \cdot |A| \cdot |x| + 2^{-106} \cdot |x|$.

G_i is a contraction: at each step, one gets a more accurate result.
g_i indicates the limit accuracy: here twice the computing precision.

Increasing the computing precision for x theoretical results

Let us iterate this refinement: $x_{i+1} = x_i + \mathrm{mid}(\mathbf{e_i})$.

Following **Higham**, one can prove that:

$$|x - x_{i+1}| = G_i |x - x_i| + g_i$$

where
$G_i \leq 2^{-53} \cdot |A^{-1}| \cdot W + 2 \cdot 2^{-53} \cdot (I + 2^{-53} \cdot |A^{-1}| \cdot W) \cdot |A^{-1}| \cdot |A|$
and
$g_i \leq 2n \cdot 2^{-106}(I + 2^{-53} \cdot |A^{-1}| \cdot W) \cdot |A^{-1}| \cdot |A| \cdot |x| + 2^{-106} \cdot |x|$.

G_i is a contraction: at each step, one gets a more accurate result.
g_i indicates the limit accuracy: here twice the computing precision.

Agenda

Conclusion

Summary:

- iterative refinement: when the problem is close to linear, the error is a solution to a problem similar to the original one;
- solving this problem allows to correct the computed solution and iterating the refinement allows to get the maximal accuracy;
- interval analysis makes it possible to **verify** the computed solution.

Conclusion and future work

What remains to be done:

- implement all these techniques;
- understand how more efficient techniques relate (or not) to iterative refinement;
- check the pen-and-paper proof using a proof-checker.

We (numerical analysts, computer arithmeticians) need you (experts in theorem-proving).

Maybe you (experts in theorem-proving) need us (numerical analysts, computer arithmeticians) to establish the proofs in the first place?

Conclusion and future work

What remains to be done:

- implement all these techniques;
- understand how more efficient techniques relate (or not) to iterative refinement;
- check the pen-and-paper proof using a proof-checker.

We (numerical analysts, computer arithmeticians) need you (experts in theorem-proving).

Maybe you (experts in theorem-proving) need us (numerical analysts, computer arithmeticians) to establish the proofs in the first place?

Conclusion and future work

What remains to be done:

- implement all these techniques;
- understand how more efficient techniques relate (or not) to iterative refinement;
- check the pen-and-paper proof using a proof-checker.

We (numerical analysts, computer arithmeticians) need you (experts in theorem-proving).

Maybe you (experts in theorem-proving) need us (numerical analysts, computer arithmeticians) to establish the proofs in the first place?